TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Tvorba pracovních plánů procesu repasování dílů v automobilovém průmyslu

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Tvrdík Aleš**
Vedoucí práce: Ing. Tomáš Martinec, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Job planning in the remanufacturing process in automotive

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Tvrdík Aleš**
Supervisor: Ing. Tomáš Martinec, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Aleš Tvrdík**
Osobní číslo:	**M13000206**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Tvorba pracovních plánů procesu repasování dílů v automobilovém průmyslu**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s fungováním jednotlivých testovacích a repasovacích linek ve společnosti TRW Frýdlant s.r.o a proveďte rešerši ukládaných dat.
2. Na základě rešerše navrhněte množinu univerzálních vstupů grafického uživatelského rozhraní pokrývající všechny typy ukládaných dat a nově zaváděný mechanismus skladového hospodářství.
3. Vytvořte aplikaci pro řízení linek repasování, která bude umožňovat i vytváření a editaci průběhu procesu pro různé druhy sestav z jednotlivých pracovních stanic a která bude využívat principu Rapid Application Development (RAD).
4. Pomocí výsledné aplikace vytvořte ukázkové procesy pro zvolené sestavy a prezentujte funkčnost ve spojení s programem pro sledování procesu repasování.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40–50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **TVRDÍK, Aleš. Sledování výrobního procesu repasování. Liberec, 2013. Bakalářská práce. Technická univerzita v Liberci. Vedoucí práce Ing. Tomáš Martinec, Ph.D.**

[2] **TRW AUTOMOTIVE AFTERMARKET. Aftermarket car parts [online]. 2013 [cit. 2013-04-18]. Dostupné z: http://www.trwaftermarket.com/en/.**

[3] **Jim Keogh, Java bez předchozích znalostí, Computer Press, 2013**

[4] **Dokumentace jazyka SQL**

Vedoucí diplomové práce:	**Ing. Tomáš Martinec, Ph.D.** Ústav mechatroniky a technické informatiky
Konzultant diplomové práce:	**Ing. Tomáš Prokeš** TRW Automotive Czech s.r.o.
Datum zadání diplomové práce:	**10. října 2015**
Termín odevzdání diplomové práce:	**16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci
L.S.
14

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 29.9.2016

Podpis:

Poděkování

V první řadě bych rád poděkoval mé rodině za morální podporu při vytváření diplomové práce i během celého studia. Dále bych chtěl poděkovat mému vedoucímu Ing. Tomáši Martincovi, Ph.D. za rady při vytváření této zprávy a kolegům z firmy TRW Frýdlant s.r.o za rady a připomínky při návrhu výsledné aplikace.

Abstrakt

Diplomová práce volně navazuje na bakalářskou práci „Sledování výrobního procesu repasování“ a na ní navazující diplomový projekt „Rozšíření stávajícího programu pro řízení procesu repase“. Stejně jako výše zmíněné práce vznikla ve spolupráci s firmou TRW Frýdlant s.r.o. Využívá poznatků získaných během nasazení původní aplikace pro sběr dat v reálném provozu a odstraňuje zjištěné nedostatky, pramenící zejména z jednoúčelovosti původního řešení. Cílem diplomové práce je vytvořit ucelený systém, který umožňuje vytvářet a spravovat uživatelské rozhraní pro jednotlivé repasovací stanice, spojovat tyto stanice do repasních linek a využívat těchto stanic pro ukládání potřebných dat. Při vytváření zmíněného systému byl kladen důraz zejména na univerzálnost a jednoduchost použití, proto je veškerá konfigurace systému prováděna prostřednictvím grafického rozhraní pomocí formulářů a implementovaného WYSIWYG designeru. Ten umožňuje umisťovat ovládací prvky v reálném čase přetažením z palety dostupných prvků podobně jako u RAD nástrojů pro vytváření grafického uživatelského rozhraní ve vývojových prostředích. Současně byla navržena nová databáze, uchovávající jak ukládaná data, tak i nastavení jednotlivých stanic a linek. Ta je přizpůsobena nově plánovanému skladovému hospodářství, ve kterém se repasované sestavy rozpadají na jednotlivé komponenty.

Klíčová slova: TRW, Repasování, Automobilový průmysl, archivace dat, grafické uživatelské rozhraní.

Abstract

This diploma thesis follows up the bachelor thesis „Monitoring of remanufacturing process“ and project called „Extension of current application for monitoring of remanufacturing process“. Both were created in cooperation with company TRW Frýdlant s.r.o. It uses experience collected during the deployment in production and eliminates drawbacks caused by uniqueness of previous solution. Target of this thesis is develop a complex system that allows create and edit user interface of work stations, group stations into remanufacturing lines and save measured data from process using created stations. Great emphasis was on versatility and simplicity of usage. According to this all system configuration can be done through a graphical interface using from and developed WYSIWYG editor. It allows to place controls in real time by draging from pallete of components as simple as in RAD tools focused on making graphical user interfaces in integrated development environments. Together were designed new database storing measured data as well as configuration of work stations and remanufacturing lines. New database is designed considering to planned warehouse management system based on division of composite into components.

Keywords: TRW, remanufacturing, automotive industry, data archiving, graphical user interface

Obsah

Seznam obrázků

Seznam symbolů a zkratek

API Application Programming Interface

BLDC Brushless Direct Current

CAN Controller Area Network

CASL Calculation and Scripting Language

ECT ECU Calibration Tool

ECU Electronic Control Unit

EEPROM Electrically Erasable Programmable Read-Only Memory

EOL „End of line“, jinými slovy koncový nebo finální

EPHS Electrically Powered Hydraulic Steering

EPS Electric Powered Steering

ER Entity-Relationship

ESD Electrostatic Dissipative

FDT Frýdlant Diagnostic Tool

GPIO General-purpose input/output

GUI Graphical User Interface (grafické uživatelské rozhraní)

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JSON JavaScript Object Notation

PLC Programmable Logic Controller

PWM Pulse-Width Modulation

RAD Rapid application developement

RTF Rich Text Format

SQL Structured Query Language

TRW počáteční písmena jmen Thompson Ramo Wooldridge, zakladatelů společnosti

URL Uniform Resource Locator

WYSIWYG akronym anglické věty „What you see is what you get“ (výstup editoru bude vypadat přesně tak, jak jej uživatel vidí na obrazovce)

XML Extensible Markup Language

XSD XML Schema Definition

1 Úvod

Tato diplomová práce volně navazuje na bakalářskou práci „Sledování výrobního procesu repasování“[1] a na ní navazující diplomový projekt „Rozšíření stávajícího programu pro řízení procesu repase“. Jejím tématem – stejně jako u předchozích prací – je maximalizovat možnosti ukládání dat získaných během procesu repasování s co nejmenším časovým zatížením operátorů. Cílem je tedy vytvořit takový systém, který operátorům umožňuje zadávat data naměřená a získaná během celého procesu a který by byl zároveň uživatelsky přívětivý a nenáročný jak pro operátory na lince, tak i pro techniky, kteří mají na starosti správu samotného procesu.

V prvních kapitolách je čtenář seznámen s repasovacím procesem v závodě TRW Frýdlant s.r.o a jsou zde vysvětleny základní pojmy používané pro jeho popis. Pro snazší pochopení je proces repasování popisován na konkrétním výrobku, kterým je elektrické čerpadlo hydraulického posilovače řízení (neboli EPHS pumpa) použitá ve vozech značky Mini, spadající od roku 1996 do skupiny BWM Group[2]. Jsou zde popsány prostředky, kterými lze data získávat, stejně jako důvody pro jejich archivaci.

V následující kapitole je vysvětlen hlavní důvod vzniku této práce: změna skladového hospodářství na lince EPS, pro kterou byla vytvořena aplikace v rámci bakalářské práce. Jsou zde popsány oba přístupy – zachování K-kitů i rozpad na jednotlivé komponenty – a porovnány výhody a nevýhody obou řešení. Sjednocení způsobu skladování na více linkách umožňuje také rozšíření vytvořené aplikace i na další linky, což se jeví jako hlavní přínos nové aplikace.

Další kapitoly již pojednávají o samotné implementaci programu vytvořeného v rámci diplomové práce. Nejprve jsou popsány úpravy provedené v návrhu SQL databáze, pramenící jak ze zkušeností z předchozí verze, tak i z plánovaného rozšíření.

Poté je popsána implementace editoru stanic, díky kterému byla původní apli-

kace rozšířena o snadnou možnost využití na více linkách. Je zde popis ovládání uživatelského rozhraní i struktury samotného programu. Naleznete zde také popis základních komponent, které je možné v editoru použít. Pro případ, že základní komponenty nebudou splňovat potřeby uživatele, je součástí této kapitoly také návod pro vytvoření vlastních komponent s popisem tříd a metod, které je nutné implementovat.

Předposlední kapitola zobrazuje konkrétní možnosti použití výsledné aplikace. Popisuje postup pro vytvoření prvních stanic, sestavení nové repasní linky a vytvoření reference pro založenou linku. Vše je vytvořeno na příkladu již zmíněného produktu - hydraulické pumpy vozu BMW Mini, která se v obchodech nachází pod označením JER137.

V závěrečné kapitole jsou shrnuty výsledky této práce a jsou zde uvedena další možná vylepšení aplikace.

2 Repasování v závodě TRW Frýdlant s.r.o

Celý proces repasování, od přijetí starých a nefunkčních dílů až po odeslání krabice s repasovaným dílem koncovému zákazníkovi, je rozložen do několika kroků a využívá mnoho interních pojmů. Cílem následující kapitoly je tento proces popsat a vysvětlit základní pojmy používané v závodě TRW Frýdlant s.r.o.

Na počátku repasování vstupuje do procesu nefunkční nebo jinak znehodnocený díl. Ten je označován v angličtině slovem „core“, a toto označení se v nezměněné formě využívá i ve frýdlantském závodě. Jde o běžně používané označení, které využívá např. i společnost Catterpillar.[3] Největším dodavatelem core jsou sami zákazníci, tj. prodejci náhradních dílů. Ti obvykle k ceně zboží připočítávají vratnou zálohu, která tvoří čtvrtinu až třetinu ceny pro koncového zákazníka. Ta je zákazníkovi navrácena při odevzdání identického dílu a po posouzení jeho stavu. Vrácený díl musí splňovat požadavky definované dokumentem „Požadavky pro vratné díly TRW“[4]. Dalším zdrojem core mohou být např. autovrakoviště. Navrácený core je po přijetí roztříděn a uskladněn.

Každý díl v procesu (označovaný jako „komponenta“) má unikátní identifikátor - číslo komponenty. Ten neslouží pouze jako typové označení, ale obsahuje i informaci o stavu repasovacího procesu. Kupříkladu hydraulické čerpadlo posilovače řízení vozu BMW Mini, nacházející se v core, má označení 8905003K. Po rozdělení na hydraulickou část a motor s řídící jednotkou neboli ECU, zrepasování obou částí a následném složení již pumpa spadá pod položku 8905003R.

Zařízení, které vstupuje do procesu (hydraulická pumpa, hydraulické/elektrické řízení apod.) je obvykle komplexní systém, který lze rozložit na více částí. V následujícím textu se díky tomu můžete setkat také s označením „sestava“ a „podsestava“. Podsestava je jiné označení pro komponentu, která je dále dělitelná. Je složena z dalších komponent, které mohou či nemusejí být podsestavou. Jako sestavu označujeme

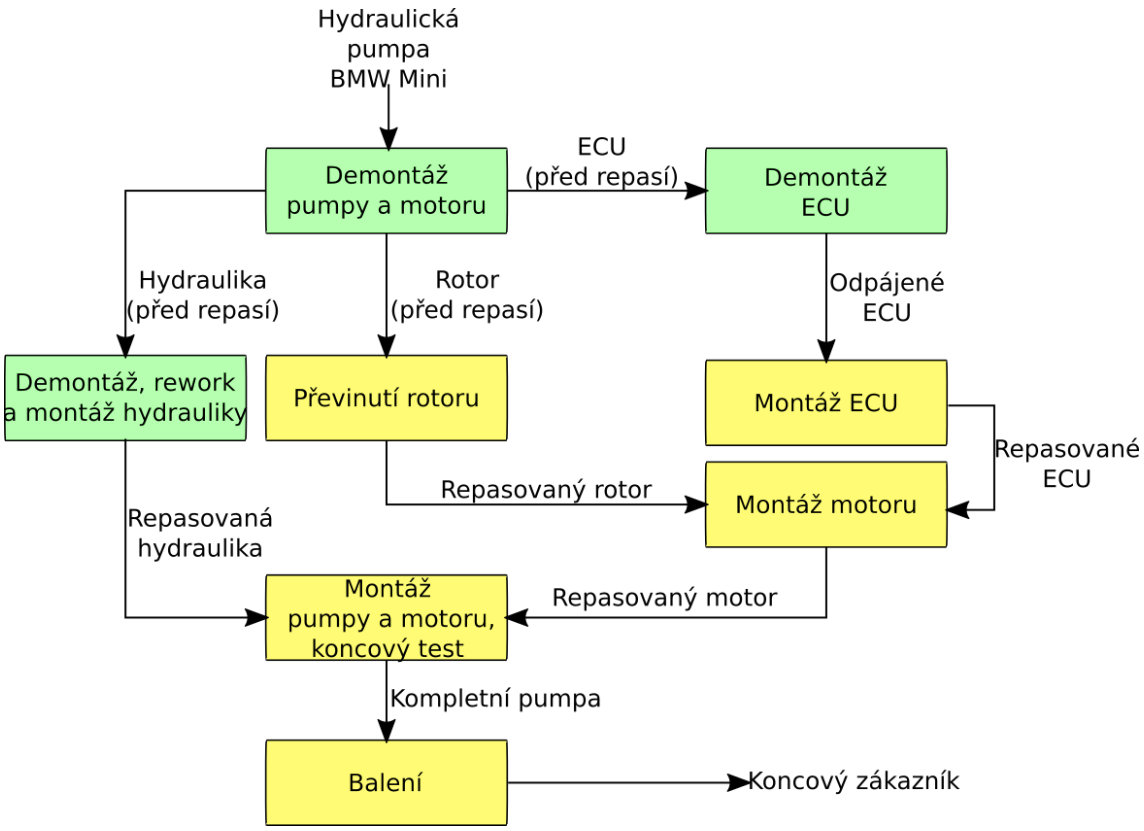
podsestavu, která se nevyskytuje v žádné další podsestavě. Pro lepší pochopení si můžeme představit kořenový strom z teorie grafů. Sestava reprezentuje kořen stromu, podsestavou je každý uzel, který má minimálně jednoho předka a minimálně jednoho potomka. Každý list stromu je poté dále nedělitelnou komponentou.

Stejně jako jednotlivé výrobky je i celý proces rozdělen do několika kroků, v rámci kterých dochází k demontáži jednotlivých dílů, jejich postupné repasi a opětovnému složení. Každý takový krok začíná vytvořením dílenské zakázky v systému. Zakázka obsahuje číslo zakázky, identifikátor typu výrobku, který je nositelem zakázky a počet kusů, jež se mají vyrobit. Dále obsahuje seznam všech komponent použitých v té části procesu, který zakázka pokrývá. Jedná se o komponenty určené k uskladnění, které byly v rámci procesu demontovány z původního dílu, nebo o komponenty, které do procesu vstupují a ze kterých je vytvořen nový (repasovaný) díl. Díky tomu můžeme rozdělit zakázky na montážní a demontážní.

Číslo zakázky je šestimístné číslo, které se inkrementuje s každým vytvořením nové zakázky. Po dosažení maximální hodnoty je počítáno od čísla 100 000. Jeden tento cyklus trvá cca 3 až 4 roky.

Identifikátor komponenty uvedený na zakázce označuje komponentu, která je výstupem zakázky. V případě demontážní zakázky je zde uvedeno číslo jedné z výstupních komponent. U montážní zakázky je naopak uvedeno číslo komponenty, která vznikne sestavením komponent vstupujících do zakázky.

Rozdělení procesu pro zmíněnou hydraulickou pumpu posilovače řízení vozu BMW Mini lze vidět na obrázku 1. Každý blok diagramu reprezentuje jednu část procesu, pro kterou je vytvořena dílenská zakázka.



Obrázek 1: Rozdělení repasního procesu pro hydraulickou pumpu řízení BMW Mini. Zelená - demontážní zakázka, Žlutá - montážní zakázka

Část procesu ohraničená jednou dílenskou zakázkou je prováděna na jedné repasní lince. Repasní linka je rozdělena do několika stanic. Stanicí rozumíme jedno pracoviště, které je vybavené nástroji a zařízeními potřebnými k provedení ucelené části procesu (nářadí, PC s diagnostickými nástroji apod.). Příkladem mohou být stanice vstupní diagnostiky, mytí, pájení nebo koncového (EOL) testu. Ukázku konkrétní stanice lze vidět na obrázku 2.

V rámci jedné stanice se provádějí konkrétní kroky (operace) repasovacího procesu dle pracovního postupu vytvořeného procesním technikem. Pracovní postup je závislý na probíhající zakázce (respektive referenci, pro kterou je zakázka založena). Mezi operace patří např. utažení šroubu na požadovaný moment, spuštění diagnostického programu, pájení/odpájení dané komponenty a podobně. Některé operace mají výstup v podobě dat, které má smysl archivovat z důvodu zpětné sledovatelnosti. Více o získaných datech lze nalézt v následující kapitole „Data získaná v průběhu repasování“.

Obrázek 2: Ukázka stanice

Výše napsané informace jsou společné pro všechny produkty. Jelikož se ale jednotlivé výrobky liší, je odlišný i konkrétní proces repasování.

Všechny výrobky je možné rozdělit dle konstrukčního typu, který určuje o jaký druh výrobku se jedná. Pod jeden konstrukční typ tedy spadají výrobky, které jsou si podobné funkcí, technickým řešením i samotnou fyzickou konstrukcí. Výrobky jednoho konstrukčního typu jsou zpravidla repasovány na jedné repasní lince. To však opačně neplatí – na jedné lince může být repasováno více konstrukčních typů. Příkladem je přehrávání řídících jednotek airbagů, které je prováděno střídavě na lince EPS a ECU. Důvod je prostý: díky tomu, že zařízení pro přehrávání je poměrně snadno přenosné a objemy výroby se pohybují řádově v jednotkách tisíc ročně se využívá ta linka, na které právě neprobíhá výroba. Existující konstrukční typy lze nalézt v dokumentu „Matice konstrukčních typů“ [5].

I mezi výrobky spadajícími pod jeden konstrukční typ jsou však jisté rozdíly. Některé výrobky neprocházejí všemi stanicemi umístěnými na lince a jejich pracovní

postupy (a tedy i prováděné operace) se mohou značně lišit, což je nutné zohlednit i ve výsledné aplikaci. Příkladem může být rozdílnost konkrétního modelu sloupku řízení pro automobily s levostranným a pravostranným řízením. Ačkoliv oba sloupky používají stejné ECU, verzi softwaru i softwarové nastavení (tj. mapy posilovacího účinku), částečně se liší např. umístěním montážních bodů na těle posilovače nebo umístěním konektorů.

3 Data získaná v průběhu repasování

Během repasovacího procesu je možné nashromáždit velké množství dat. Ta mohou sloužit např. při reklamaci konkrétního kusu ke zjištění příčiny závady. Lze kupříkladu částečně zkontrolovat dodržení pracovního postupu a tím vyloučit chybu operátora. Další možností je kontrola, zdali byl stejný problém zjištěn před začátkem repasování. V takovém případě lze předpokládat, že příčina selhání nebyla během repasování odstraněna, a tak může být reklamovaný díl předán technikům k detailnější diagnostice za účelem zlepšení repasního procesu. V neposlední řadě je možné zjistit, zdali je příčinou komponenta, která byla během repasování vyměněna. Pokud příčinou selhání byla nová komponenta, je možné ji reklamovat u dodavatele, případně současného dodavatele vyměnit za nového, který je schopen dodávat komponenty požadované kvality.

Další možnost využití nasbíraných dat je pro dlouhodobou analýzu a následnou úpravu repasovacího procesu z důvodu zvýšení výtěžnosti a snížení počtu reklamací. V případě, že je většina závad způsobena například nefunkční komponentou na desce ECU, může být z finančního pohledu výhodnější zahrnout do procesu testování těchto komponent, případně je pokaždé automaticky měnit za nové kusy. I tak může celková cena za komponenty v součtu s finančním ohodnocením času operátora, který stráví odpájením staré komponenty a připájením nového kusu, být pouze zlomek ceny, kterou by stály případné reklamace. Pro tyto účely se obvykle využívá Paretova analýza, která vychází z obecně známého „pravidla 80/20“ (80% reklamací bylo způsobeno 20% možných vad). [6]

Nasbíraná data mohou obsahovat záznamy o chybějících či poškozených komponentách, nebo o komponentách, které byly nahrazeny během procesu. Po úpravě stávajícího vybavení je také možné ukládat hodnoty momentů při utahování šroubových spojů programovatelnou utahovačkou řízenou pomocí PLC, případně hodnoty

pro vykreslení grafů tlaku a průtoku na EOL testovačce hydraulických pump. Je vhodné také zaznamenávat ručně naměřené hodnoty (např. test sklonu a dosahu u EPS) nebo další veličiny (hlučnost). Mezi nejvýznamnější data však patří výstupy speciálních programů používaných pro diagnostiku ECU nebo při funkčním testování. Jako příklad si můžeme uvést tři nejčastěji používané programy: **ECT**, **FDT** a **CANape**.

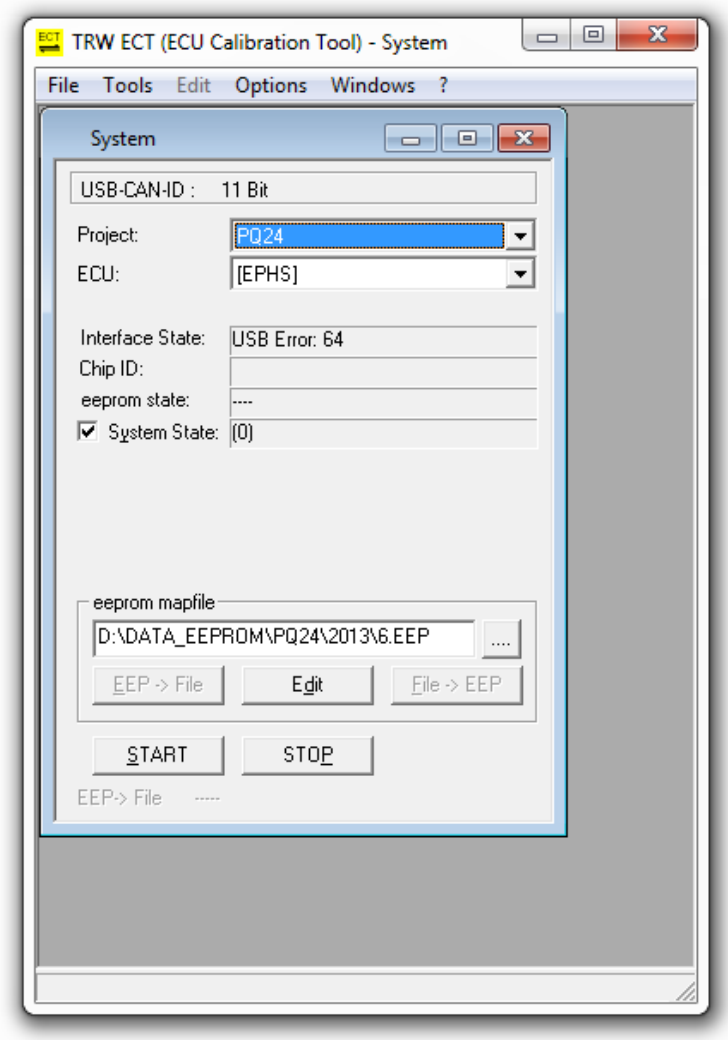
ECT (neboli ECU Calibration Tool) je software vytvořený vývojovým centrem společnosti TRW, které sídlí v německém Düsseldorfu. Používá se zejména pro diagnostiku a testování EPHS pump. Umožňuje načítat hodnoty (identifikační data výrobce, verze softwaru, zaznamenané chyby apod.) z EEPROM řídící jednotky, exportovat získaná data do textových souborů a mazat paměť chyb. Tento software pro komunikaci s pumpou využívá speciální hardwarové zařízení – starterbox – zobrazené na obrázku 3.

Obrázek 3: Zařízení pro ovládání EPHS pump pomocí softwaru ECT

Pro diagnostiku a ovládání připojené pumpy je v první řadě vybrán výrobce a typ pumpy. Software poté nahraje do starterboxu CAN zprávy ve formátu požadovaném připojenou pumpou. Ty simulují CAN komunikaci v automobilu, zejména zprávy nutné pro bezchybný chod pumpy. Ovládání chování pumpy (roztočení mo-

toru pumpy, regulace otáček) je realizováno pomocí ovládacích prvků na starterboxu.

V současné době se pracuje na implementaci všech funkcí ECT do programu FDT, aby jej mohl v budoucnu plně nahradit.

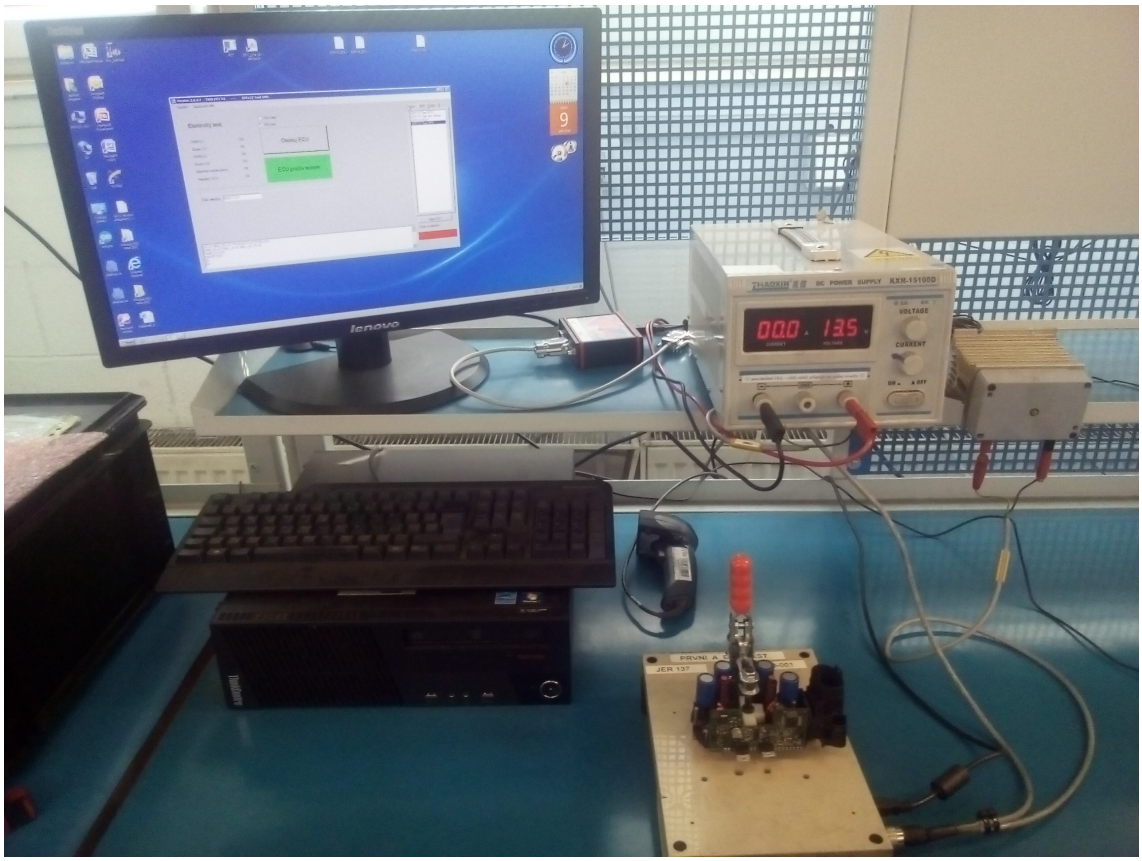


Obrázek 4: Hlavní okno programu ECT

FDT, neboli Frýdlant Diagnostic Tool, je software vyvíjený přímo ve frýdlantském závodě pro účely diagnostiky a testování. Je naprogramovaný v jazyce C# na platformě .NET a využívá API a hardwarové řešení třetích stran pro komunikaci a ovládání testovaného zařízení.

Pro interakci s moderními zařízeními, které komunikují po sběrnici CAN, využívá zařízení CanCaseXL společnosti Vector ve spojení s API Vector XL Driver Library. Ke komunikaci přes sběrnici K-Line využívá VAG KKL kabel a standardní sériovou

linku. Poslední možností interakce s testovaným zařízením je pomocí GPIO pinů, díky kterým je možné ovládat analogové veličiny. K tomuto účelu je využita měřící karta společnosti National Instruments a příslušné API. Lze tak simulovat úbytek napětí na termistorech pro otestování správného přepočtu na teplotu v řídící jednotce, generovat PWM pro otevírání budících tranzistorů či simulaci magnetických senzorů, měřit hodnoty odporů na desce plošných spojů apod. Většinu podobných měření je možné provádět pouze pomocí testovacích bodů přímo na desce plošného spoje. Z tohoto důvodu jsou vytvářeny testovací stolice umožňující snadné založení testovaného obvodu a jeho komplexní test spuštěný v programu FDT. Příklad takového zařízení můžete vidět na obrázku 5.

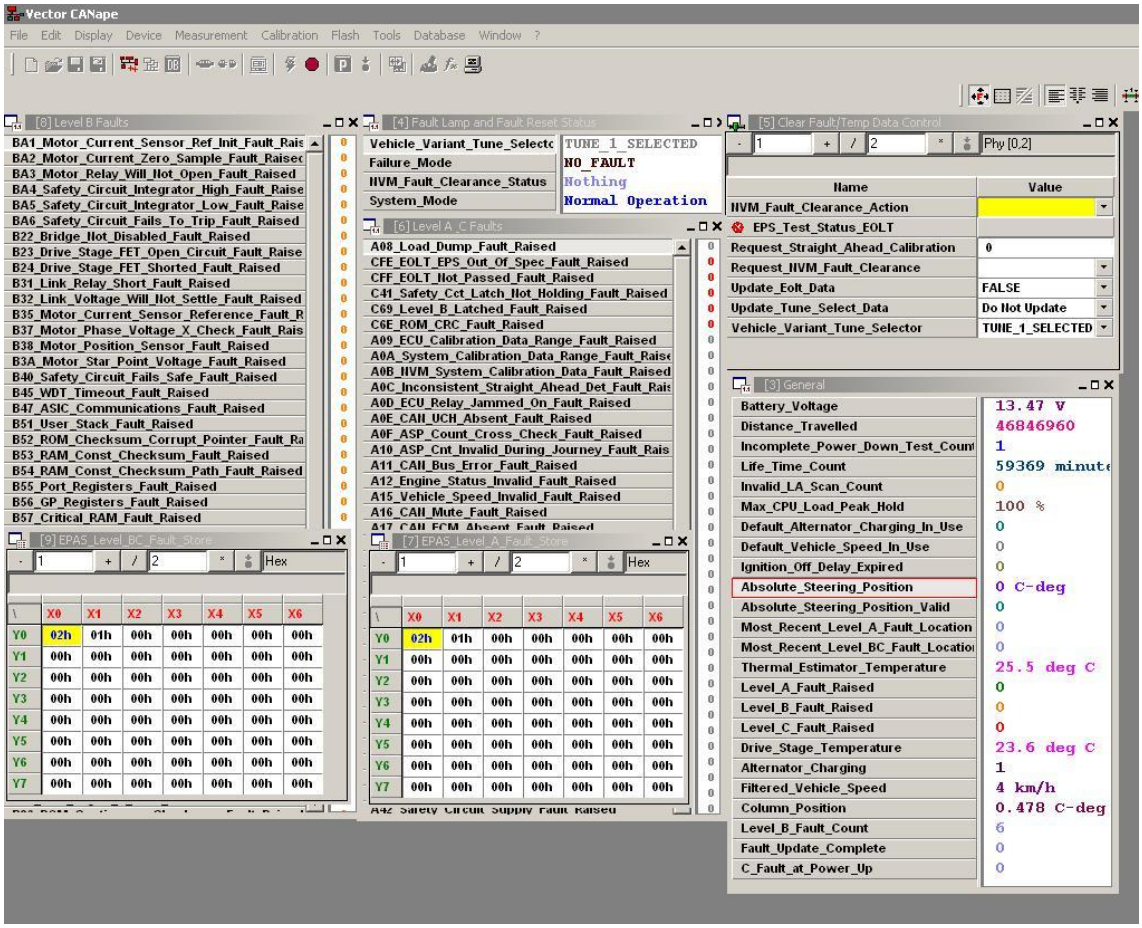


Obrázek 5: Ukázka testování pomocí programu FDT a testovacího zařízení (testování řídící jednotky hydraulické pumpy BMW Mini)

Ačkoliv je v současné době možné spouštět test pouze z grafického uživatelského rozhraní, pracuje se na rozšíření programu FDT o možnost spuštění testu prostřednictvím příkazové řádky. Pomocí parametrů příkazu lze vybrat požadovaný test,

který je automaticky spuštěn. Výsledek testu a naměřená data jsou poté vypsány na standardní výstup.

CANape je komerční software vytvořený německou firmou Vector Informatik. Jde o komplexní nástroj pro měření, kalibraci a diagnostiku všech úloh prováděných elektronickou řídící jednotkou. Umožňuje měřit a kalibrovat vnitřní parametry ECU, analyzovat naměřená data a převádět je do uživatelsky přívětivého formátu (vykreslování posilovacích křivek apod.) nebo přehrávat ovládací software ECU (tzv. flashování)[7]. Je používán výhradně pro diagnostiku a testování jednotek ve sloup-



Obrázek 6: Diagnostika řídící jednotky pomocí programu CANape

cích řízení na lince EPS. Krom přímé editace parametrů ECU pomocí grafického rozhraní umožňuje CANape vytváření skriptů pomocí skriptovacího jazyka CASL, jehož syntaxe vychází z jazyka C[8]. V případě, kdy možnosti vestavěného skriptovacího jazyka nedostačují a je potřeba plně automatizované ovládání externí aplikací, je možné využít rozhraní CANape API. To poskytuje ostatním aplikacím služby

pro přístup do připojené ECU prostřednictvím programu CANape, který v tomto případě běží jako server a předává klientské aplikaci naměřené hodnoty. [9] CANape komunikuje se řídící jednotkou po sběrnici CAN pomocí hardwarových zařízení CanCaseXL nebo CanBoardXL společnosti Vector.

4 Změna skladového hospodářství na lince EPS

Jedním z důvodů vzniku této diplomové práce je plánovaná změna způsobu uskladnění komponent během procesu za účelem sjednocení způsobu skladování dílů na více linkách.

Na lince EPS, pro kterou byla vytvořena původní aplikace, jsou díly uchovávány v takzvaných K-kitech. Příchozí core je v rámci demontážní zakázky diagnostikován pro zjištění charakteru závady, dále je očištěn a demontován na jednotlivé komponenty. Vadné komponenty spolu s komponentami, které se standardně vyměňují za nové, jsou vyhozeny. Zbylé komponenty jsou uloženy do ESD boxu, čímž vzniká K-Kit. Ten je poté uložen do skladu. Výhodou tohoto systému je, že všechny komponenty z původního kusu zůstávají pohromadě. Lze je tedy při následné montážní zakázce spojit s proběhlou demontážní zakázkou a se všemi daty, které byly během demontáže získány. Nevýhodou je prostorová náročnost při naskladnění. Další nevýhodou je, že ačkoliv lze snadno zjistit počet uskladněných K-kitů, je obtížné zjistit počet jednotlivých komponent. Ta je částečně odstraněna díky původní aplikaci, která umožňuje vhodně zvoleným SQL dotazem nad použitou databází zjistit počty jednotlivých komponent. Vyvstává ale další problém, který uvedu na příkladu:

Při založení montážní zakázky na 100 kusů posilovače je nutné vyskladnit 100 kusů K-Kitu. Vzhledem k tomu, že ale K-Kit nemusí obsahovat všechny komponenty (pokud byla nějaká komponenta z původního kusu nepoužitelná), je nutné vyskladnit zvlášť tyto chybějící komponenty. To lze udělat dvěma způsoby. První možností je postupně vyskladňovat komponenty tak, jak je potřeba. To však zdržuje samotný proces repasování čekáním na potřebné díly, a není logisticky výhodné dopravovat komponenty ze skladovacích prostor do prostoru výrobní linky po jedné. Druhou

možností je zjistit počet komponent, které je nutné vyskladnit navíc před zahájením repasování ze zaznamenaných dat. To však znamená vyhledat data postupně o každém vyskladněném K-kitu, a vyskladnit komponenty, které byly uvedeny jako chybějící. To je v případě více kusů časově náročné. Určitým řešením se může jevit doplnění chybějících komponent do K-Kitu už během demontážní zakázky. Tím je ale problém vyskladnění komponent pouze přesunut do jiné části procesu, neboť nelze předem znát počty komponent, které budou během demontáže vyřazeny.

Z těchto důvodů se jako výhodnější řešení může jevit uskladňování samotných komponent. Výhodou tohoto řešení je ušetření skladových prostor a snadný přehled o počtech uskladněných komponent. Při vyskladňování komponent pro montážní zakázku je počet komponent přesně daný počtem kusů, které se mají v dané zakázce vyrobit. Tento systém je použitý na ostatních linkách a jeho nasazení je plánováno i pro zmíněnou linku elektrických sloupků řízení. Nevýhodou tohoto systému však je, že nelze zpětně dohledat původ dané komponenty a tím i diagnostická data a další informace o kusu, ze kterého byla komponenta demontována.

Tento problém má za úkol vyřešit aplikace vytvořená v rámci diplomové práce. Většina komponent, u kterých je zajímavé udržovat zpětnou vyhledatelnost (ECU, motory, senzory), obsahuje čárové kódy nebo 2D kódy, které je jednoznačně identifikují. Problém nastává v případě, že komponenta neobsahuje jedinečný identifikátor. Ať už z důvodu, že byl tento kód odstraněn nebo je nečitelný kvůli jeho poškození.

5 Návrh a implementace databáze

Původní aplikace využívala pro uchování dat MySQL databázi běžící na serveru. Použití tohoto relačního databázového systému se osvědčilo, proto jej využívá i nově vzniklá aplikace. Struktura databáze byla částečně upravena a rozšířena. ER diagram původní i upravené databáze nebylo možné kvůli svému rozsahu začlenit do textu této práce, proto je umístěn na přiloženém CD viz Příloha A: Obsah přiloženého CD.

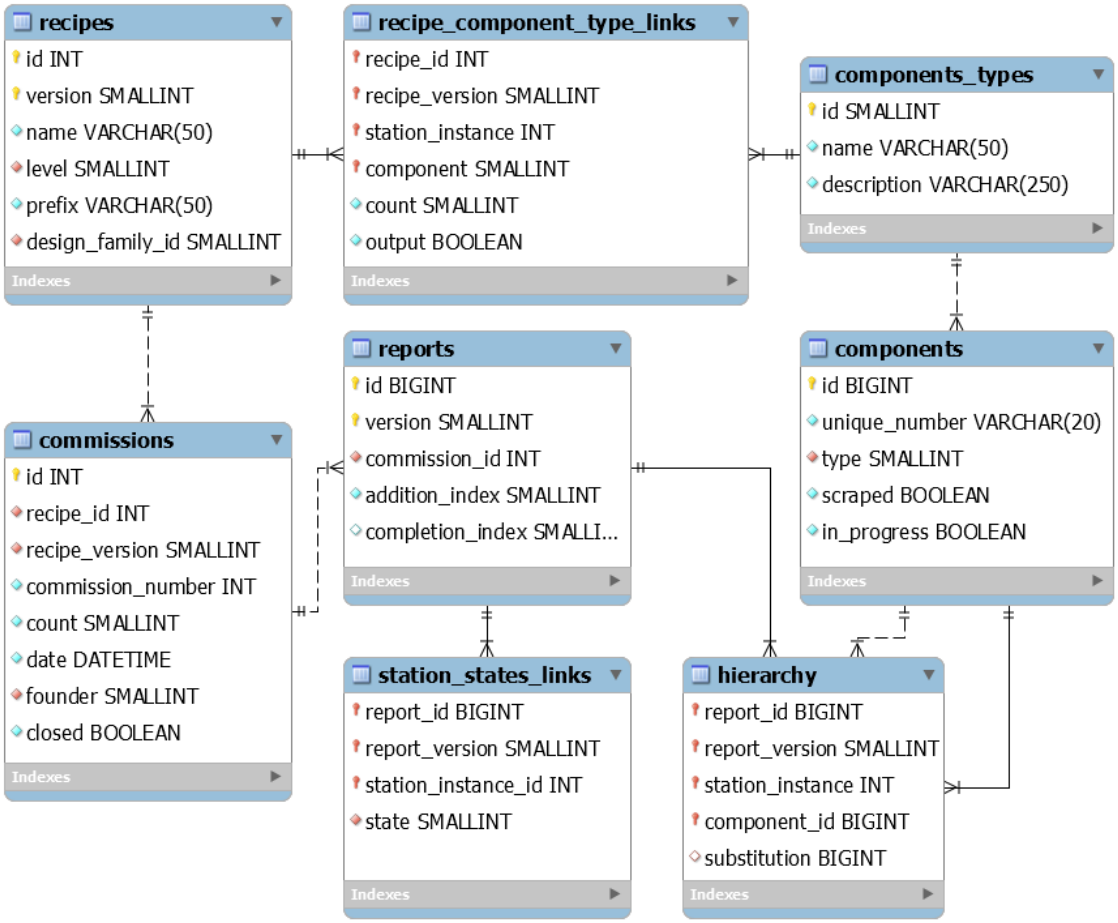
Provedené změny lze rozdělit dle jejich motivace do tří následujících podkapitol. V každé jsou detailně vysvětleny provedené úpravy a jejich důvod.

5.1 Úpravy databáze plynoucí ze změny skladového hospodářství

V původní aplikaci pro linku EPS byla veškerá data vázaná na konkrétní kus (K-Kit). Při přidání kusu do demontážní zakázky byl v databázi vygenerován záznam, jehož primárním klíčem bylo automaticky generované číslo funkcí auto_increment [10]. Toto číslo v celém procesu vystupuje jako identifikátor daného kusu a v databázi jsou k němu vázána veškerá uložená data. Při přidání kusu do montážní zakázky byl uživateli nabídnut seznam EPS, které bylo možné do dané zakázky přidat (tj. všechna EPS vytvořená v rámci demontážní zakázky, jejíž referenci lze svázat s referencí dané montážní zakázky). Výběrem uživatel určil, se kterým kusem mají být získaná data svázána.

Díky rozpadu na jednotlivé komponenty nelze tento postup aplikovat i v nové verzi programu. Získaná data není vhodné ani párovat s jednotlivými komponentami. V aplikaci totiž není jasně dané, která data náleží k jaké komponentě, a mimo to lze ukládat i data, která nemají žádnou logickou souvislost se žádnou z kompo-

nent (např. jméno operátora, který daný úkon prováděl). Z tohoto důvodu byla do hierarchie databázových objektů přidána další vrstva - tabulka reportů. Při založení zakázky a přidání nového kusu do zakázky je v databázi vygenerován záznam s unikátním identifikátorem reportu, obdobně jako v předchozí verzi. Na report jsou vázána jak data získaná během procesu, tak i identifikátory konkrétních komponent, které do reportu vstoupily nebo z něj vystoupily. Identifikátory komponent jsou vždy generovány databází. Pokud je komponenta označena unikátním kódem samotným výrobcem, lze tento kód v programu načíst a uložit. Ten jej poté využívá pro vyhledávání a podobně, ale databáze stále interně využívá svůj vygenerovaný kód, sloužící jako primární klíč. Na obrázku 7 lze vidět konkrétní implementaci tabulek, které uchovávají vazby mezi referencemi, zakázkami, reporty a komponentami. Tabulka *recipes* obsahuje základní informace o jednotlivých referencích. Vazební



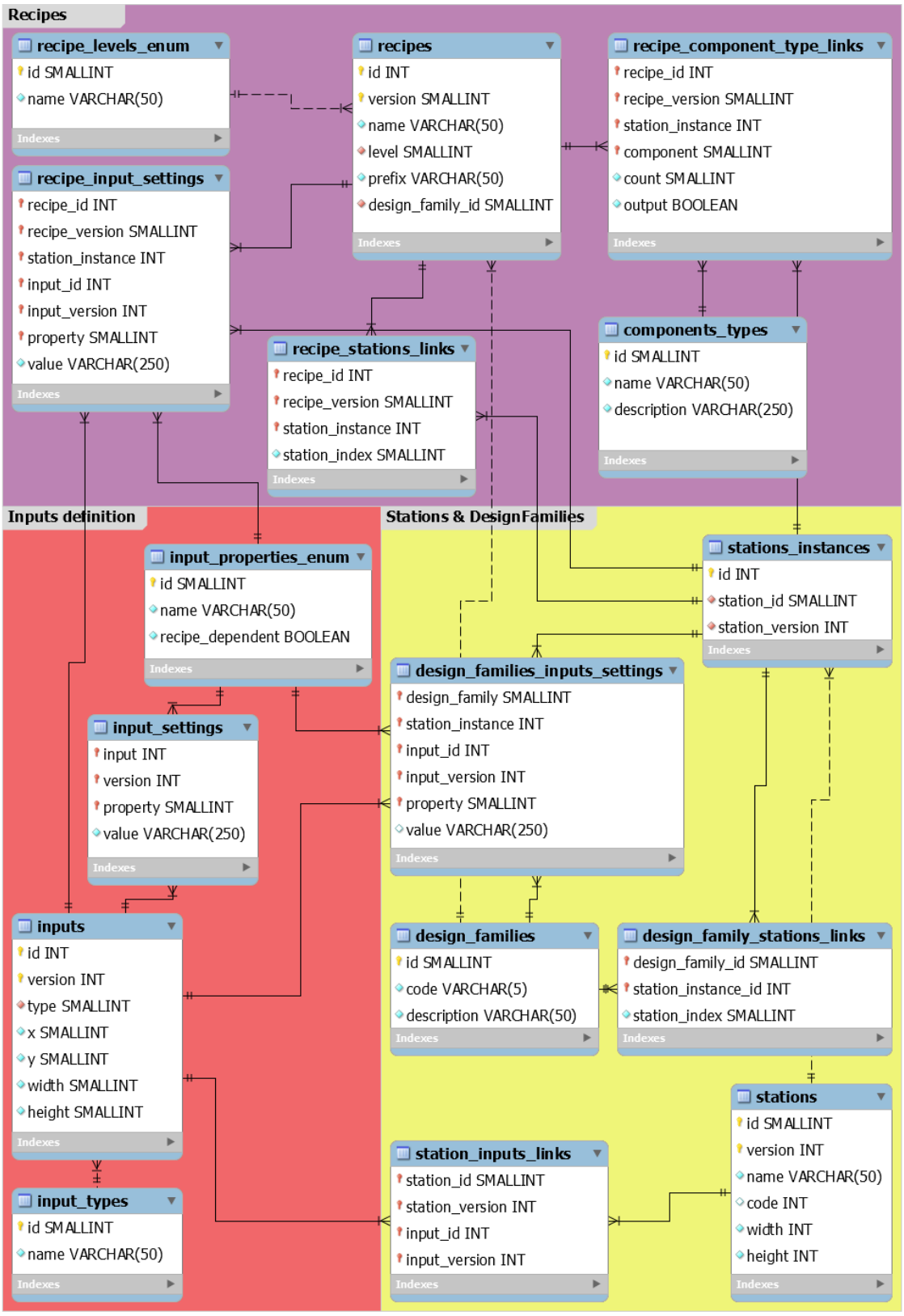
Obrázek 7: ER diagram uložení reportů v databázi

tabulka *recipe_component_type_links* určuje kolik komponent, kterého typu a na jaké stanici do procesu vstupuje, případně vystupuje. Na základě těchto vazeb generuje aplikace formulář pro zadání unikátních identifikátorů komponent. Pokud komponenta má vlastní identifikátor, je v tabulce *components* vyhledán záznam o komponentě, která je stejného typu (atribut *type*) a má stejné označení dané atributem *unique_number*. Pokud není nijak označena, je v tabulce vygenerován nový záznam a do atributu *unique_number* je zduplikován primární klíč. Tyto konkrétní komponenty jsou přiřazeny aktuálnímu reportu z tabulky*reports* skrz tabulku *hierarchy*. Vhodným složeným dotazem nad těmito tabulkami je možné zjistit, které komponenty vstupují v daném reportu nebo z něho vystupují, stejně jako čísla reportů spojená s danou komponentou. Díky tomu lze sestavit kompletní historii pro libovolnou komponentu.

5.2 Úpravy databáze nutné pro rozšíření na další linky

Aby bylo možné rozšířit program na další repasní linky, bylo nutné rozšířit databázi o mechanismus uchovávající nastavení jednotlivých repasních linek a stanic, ze kterých jsou složeny, stejně jako vyřešit přístupová práva jednotlivých uživatelů. ER diagram této části databáze najdeme na obrázku 8.

Z důvodu repasování více konstrukčních typů na jedné lince (viz kapitola č.2) nejsou v programu vytvářena nastavení pro jednotlivé linky, nýbrž pro konstrukční typy. Ke každému konstrukčnímu typu jsou krom názvu a popisku (uložených v tabulce *design_families*) přiřazeny všechny stanice, na kterých probíhá repasování. Toto párování zajišťuje vazební tabulka *design_family_stations_links*. Aby však bylo možné jednu stanici využít u více konstrukčních typů, není ve vazební tabulce použit přímo cizí klíč na definici stanice, nýbrž je vytvořen záznam v tabulce *station_instances*, jehož atributy jsou identifikátor a verze stanice, kterou daná instance reprezentuje. Díky tomu je také v jednom konstrukčním typu možné použít vícekrát stejnou stanici, avšak s různým nastavením prováděných operací.



Obrázek 8: ER diagram uložení stanic a vstupů

Samotná definice vzhledu grafického rozhraní stanice je uložena v tabulkách *stations* a *station_inputs_links*. Prvně zmíněná obsahuje obecné informace o stanici, jako jméno, kód stanice a velikost panelu, na kterém je vykreslena, pro opětovné vytvoření grafického rozhraní. Tabulka *station_inputs_links* obsahuje párování jednotlivých komponent umístěných na stanici. Většina grafických komponent tvořících panel stanice je určena pro zadávání dat. Z tohoto důvodu jména tabulek obsahují označení „inputs“.

Obecné informace o grafické komponentě zobrazené na stanici jsou uloženy v tabulce *inputs*. Jsou zde uloženy souřadnice levého horního rohu, výška a šířka pro vykreslení a typ komponenty. Možné typy komponent jsou uvedené v tabulce *input_types*. Každý typ komponenty může mít přiřazeny další vlastnosti, které určují její chování a vzhled. Mezi tyto vlastnosti patří například popisek zobrazený uživateli, kontrola formátu u textových vstupů, relevance apod. Které vlastnosti se týkají dané grafické komponenty je dáno implementací v aplikaci. Množina všech možných vlastností je uložena v tabulce *input_properties_enum* a je naplněna při vytváření databáze samotnou aplikací.

Každá vlastnost má krom generovaného primárního klíče daný název a atribut určující závislost na referenci. Vlastnosti nezávislé na referenci jsou dané již při editaci stanice, například popisek vstupu. Jiné jsou nastavovány v závislosti na referenci, jako třeba relevance. Ta udává, zda-li je nutné daný vstup vyplnit před ukončením stanice, jeho vyplnění je volitelné a nebo není v dané referenci použit.

Nastavení vlastností vstupů, které jsou nezávislé na referenci je uloženo v tabulce *input_settings*. Nastavení referenčně závislých vlastností je uloženo v tabulce *recipe_input_settings*. Program také umožňuje při vytváření konstrukčního typu navolit výchozí nastavení vstupů, které bude předvyplněno při zakládání reference. Toto nastavení se nachází v tabulce *design_families_inputs_settings*.

Všechny vlastnosti musejí být uloženy jako řetězec. Délka řetězce je omezena na 250 znaků, což by mělo plně dostačovat. Jako jistá nevýhoda se může jevit fakt, že i čistě číselné vlastnosti, jako např. maximální délka řetězce zadaná do textového vstupu, jsou ukládány jako řetězec. Jedná se ale o hodnoty, které nejsou přímo ručně editovatelné. Hodnoty vlastností jsou generovány z nastavení provedeného uživatelem, u kterého lze kontrolovat validitu. Při ukládání dochází k převodu číselné

hodnoty na text. Ten je při načtení opět automaticky převeden na číslo, nemůže tedy v programu dojít k výjimce na základě chybného formátu. Navíc lze tímto způsobem do databáze ukládat i složitější objekty pomocí serializace/deserializace, případně použít standardizovaný formát jako např. JSON. Při povolení ukládat složitější struktury (např. pole) může vyvstat otázka atomicity ukládaných dat. Splňuje v tomto případě tabulka první normální formu? Jak uvádí Christopher Date ve své knize [13] na obdobném příkladě, splňuje. Z pohledu databáze jde o dále nedělitelný řetězec jednoznačně udávající nastavení konkrétní části aplikace. Jedná se o neklíčový atribut, který není třeba indexovat kvůli vyhledávání a se kterým bude databáze vždy pracovat pouze jako s celkem. Z tohoto důvodu není nutné vytvářet strukturu tabulek umožňující ukládání různých datových typů tak, jak je používá aplikace. V opačném případě by došlo pouze ke zvýšení složitosti návrhu databáze bez navýšení výkonu či usnadnění práce s databází. V otázce atomicity ukládaných dat neexistuje jednoznačná odpověď – vždy záleží pouze na tom, jak s daty chceme pracovat[13].

V tabulkách pro nastavení stanic i vstupů je vždy použit složený primární klíč, který je tvořen automaticky generovaným identifikátorem a číslem verze. Verzování a uchovávání předchozího nastavení je důležité zejména pro udržení správné reprezentace dříve uložených dat. Důvod uvedu na příkladu:

Uvažujme referenci s jednou stanicí obsahující vstup pro zadání hlučnosti v decibelech. Aby mohl kus být úspěšně repasován, musí hodnota hlučnosti být pod stanoveným limitem. V případě, že by došlo k úpravě procesu a snížení přípustného limitu, došlo by bez verzování k situaci, kdy by již úspěšně zrepasované výrobky přestaly splňovat původní povolený limit a byly by chybně označeny jako neopravitelné, přestože byly odeslány zákazníkovi. Stejně tak by v případě rozšíření procesu o novou stanici byly všechny již zrepasované výrobky označeny jako neukončené z důvodu nedokončení nově přidané stanice.

V případě úpravy komponenty stanice a uložení se vytváří nový záznam v tabulce *inputs* se stejným atributem *id* a inkrementovaným atributem *version*. Současně je ale vytvořen nový záznam i v tabulce *stations*. Aplikace poté vytvoří nové verze všech nastavení pro reference, které obsahují upravenou stanici. Jedinou změnou oproti původním nastavením referencí je nahrazení instance stanice odkazující na

předchozí verzi stanice novou instancí již editované stanice.

Složený klíč obsahující identifikátor a číslo verze byl zvolen oproti klasickému databází generovanému id (pomocí auto_increment), použitému v předchozí verzi programu, z důvodu lepší čitelnosti a snadné vyhledatelnosti všech úprav dané komponenty/stanice/receptury.

Poslední výraznější změnou bylo přidání vazební tabulky pro provázání uživatelských účtů s konstrukčními typy, se kterými může uživatel pracovat. Při zakládání zakázky jsou uživateli nabídnuty pouze reference vycházející z konstrukčních typů přiřazených danému uživateli. Uživatel s přiřazeným oprávněním k danému konstrukčnímu typu a navíc s oprávněním spravovat konstrukční typy jej smí editovat, stejně jako nastavení referencí, které z něho vycházejí.

5.3 Úpravy plynoucí ze zkušeností z první verze

Během přibližně ročního nasazení původní aplikace v ostrém provozu bylo zjištěno několik možností jak upravit původní návrh.

První úpravou aplikace je přidání tabulky *remsys_database_properties*. Obsahuje pouze dva sloupce: *key* typu *VARCHAR(50)* a *value* typu *VARCHAR(255)*. Jak již názvy sloupců napovídají, slouží k uložení dvojic klíč-hodnota. V této tabulce mohou být uloženy obecné informace o databázi (např. číslo verze návrhu, datum vytvoření databáze apod.). Jelikož aplikace v dialogu pro výběr databáze automaticky zobrazuje všechny databáze na serveru daném síťovou adresou a portem, je vhodné odfiltrovat ty databáze, které s programem nejsou kompatibilní. To je provedeno na základě existence této tabulky. Dotaz pro získání všech databází na serveru splňující výše uvedené podmínky je uveden v ukázce č.5.1

Dotaz se skládá ze tří jednoduchých vnořených pod-dotazů. Databáze *information_schema* existuje v rámci každé instance MySQL databáze a obsahuje informace o všech ostatních databázích na daném serveru, stejně jako o serveru samotném. [14]. Dotazem *result* začínajícím na řádku č.3 jsou z této databáze získány záznamy o všech sloupcích, které náleží tabulce s názvem *remsys_database_properties*, obsahující přesně dva sloupce, jejichž název a datový typ odpovídá požadovaným hodnotám. Řádky vrácené tímto dotazem jsou již pouze seskupeny dle názvu da-

tabáze a je funkcí *count(*)* zjištěn počet sloupců, které odpovídají. Ten musí být roven celkovému počtu sloupců dané tabulky - tedy 2.

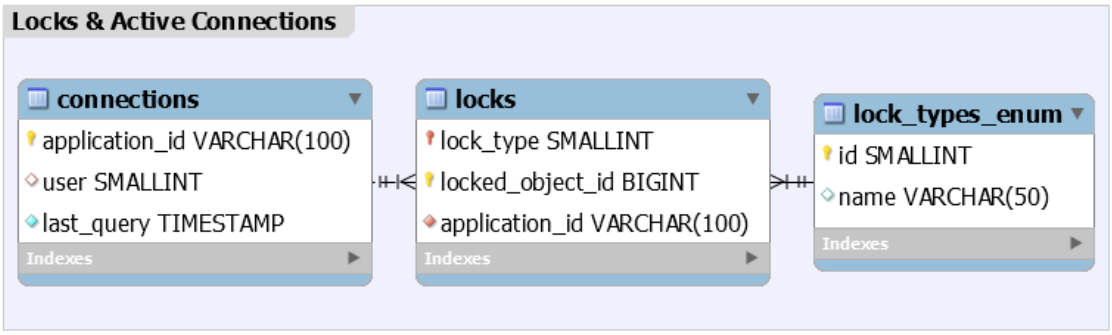
```
SELECT result.table_schema, result.table_name, result.cols_count
FROM
   (SELECT tbls.table_schema,
      (SELECT COUNT(*)
         FROM information_schema.columns cols
         WHERE
            cols.table_schema = tbls.table_schema
            AND
            cols.table_name = tbls.table_name
      ) cols_count,
      tbls.table_name,
      clms.column_name,
      clms.column_type
      FROM information_schema.tables tbls
      LEFT JOIN information_schema.columns clms ON
         (tbls.table_schema = clms.table_schema
         AND
         tbls.table_name = clms.table_name)
      WHERE tbls.table_name = 'remsys_database_properties'
      HAVING cols_count = 2
         AND (
            (clms.column_name = 'key'
             AND
             clms.column_type = 'varchar(50)'
            )OR(
             clms.column_name = 'value'
             AND
             clms.column_type = 'varchar(255)')
         )
   ) result
GROUP BY table_schema
HAVING COUNT(*) = result.cols_count;
```

Ukázka zdrojového kódu 5.1: SQL dotaz pro získání kompatibilních databází

Pokud databáze obsahuje tabulku stejného názvu se stejným formátem ukládaných dat, je kontrolována přítomnost záznamu s klíčem *DATABASE_SCHEMA_VERSION*. Ten obsahuje číslo verze databáze, které musí souhlasit s číslem uloženým v programu. Důvodem k této kontrole je neustálý vývoj aplikace. V případě, kdy dojde ke změně struktury databáze, je třeba toto číslo v programu aktu-

alizovat. Při vytváření databáze program toto číslo uloží do zmíněné tabulky. Tím zabráníme připojení aktualizované aplikace na starší verzi databáze a potencionálnímu pádu aplikace z důvodu přístupu k neexistující či pozměněné tabulce.

Druhým důležitým vylepšením je vytvoření tabulek uchovávajících zámky nad editovanými objekty. Původní aplikace obsahovala pouze tabulku s identifikátory aktuálně editovaných kusů, aby bylo zabráněno současné editaci jednoho záznamu dvěma uživateli. Nebyla však řešena současná editace nastavení reference. Z tohoto důvodu byla vytvořena struktura zobrazená na obrázku 9.



Obrázek 9: ER diagram zamykání objektů aplikace

Tabulka *connections* udržuje informace o aktuálně přihlášených uživatelích. Klíčový atribut *application_id* obsahuje název připojeného počítače. To zabraňuje spuštění dvou instancí aplikace na jednom stroji. Atribut *user* je cizí klíč do tabulky uživatelů. Atribut *last_query* obsahuje čas posledního ohlášení aplikace. Klientská aplikace po spuštění přidá záznam do tabulky *connections* a spustí vlákno, které periodicky aktualizuje atribut *last_query*. Na straně databázového serveru běží událost, která periodicky kontroluje tento časový otisk pro všechny záznamy. V případě, že dojde k pádu klientské aplikace (způsobenou výjimkou v aplikaci, resetováním počítače apod.) server zjistí, že aplikace již delší dobu neprovedla aktualizaci a tedy byla pravděpodobně ukončena. Díky tomu může uvolnit zámky (smazat příslušné záznamy z tabulky *locks*) vytvořené touto aplikací, takže nedojde k permanentnímu zablokování zamčených objektů. Tabulka *locks* obsahuje atribut udávající typ zámku, číselný identifikátor zamčeného objektu a identifikátor aplikace, která uzamčení provedla. Možné typy zámků jsou uloženy v tabulce *lock_types_enum*. Mezi typy zámků patří zámek uživatelských dat, reference, stanice, reportu apod. Při editaci objektu

je vždy proveden pokus o vložení zámku. Pakliže je vrácena výjimka, že záznam již existuje, není editace povolena a uživatel je informován prostřednictvím aplikace.

6 Implementace editoru stanic a linek

6.1 Předchozí řešení a jejich nevýhody

První verze programu vytvořená v rámci bakalářské práce obsahovala pouze stanice, které se nacházely na lince repasování elektrických sloupků řízení. Tyto stanice byly napevno definované ve zdrojovém kódu aplikace. Každá změna v pracovním postupu pro danou stanici nebo v ukládaných datech tedy znamenala úpravu zdrojového kódu a kompilaci nové verze programu. Proto i zdánlivě nepatrná úprava byla časově náročná a při každém zásahu do programu zde byla pravděpodobnost zanesení chyby do funkčního kódu. Modifikovaná verze programu musela být většinou nasazena současně na všech stanicích, aby byla dodržena kompatibilita s databází, což aktualizaci ještě více komplikovalo, neboť bylo nutné nasadit program na linku v době, kdy neprobíhala výroba.

V rámci diplomového projektu byla definice stanic vyjmuta ze zdrojového kódu do souborů XML. To umožňovalo nadefinovat stanici pomocí tagů definovaných v přiloženém souboru XSD. Aplikace poté vytvářela grafické rozhraní stanic automaticky.

Toto řešení však od uživatele vyžadovalo větší znalost práce s počítačem, zejména znalost zápisu XML dokumentů. Přestože existence XSD schématu výrazně usnadňovala vytváření souborů definujících vzhled jednotlivých stanic, bylo nutné pro úspěšné vytvoření souboru stanice znát význam jednotlivých tagů a možnosti jejich nastavení. Prvotní nastudování XSD schématu spolu s vytvořením XML souborů jednotlivých stanic bylo časově náročné. V praxi to znamenalo, že vytváření a editace stanic nemohla být prováděna přímo procesním technikem, přestože se jedná o záležitost samotného procesu.

Zároveň nebylo možné definovat přesný vzhled stanice, jelikož každý prvek měl

předem nastavenou velikost a byl na panel stanice umisťován programem. Úprava stanice také nebyla výrazně zjednodušena. Tabulky v relační databázi byly vytvořeny pouze při prvním spuštění programu.

6.2 Návrh současného řešení s grafickým editorem

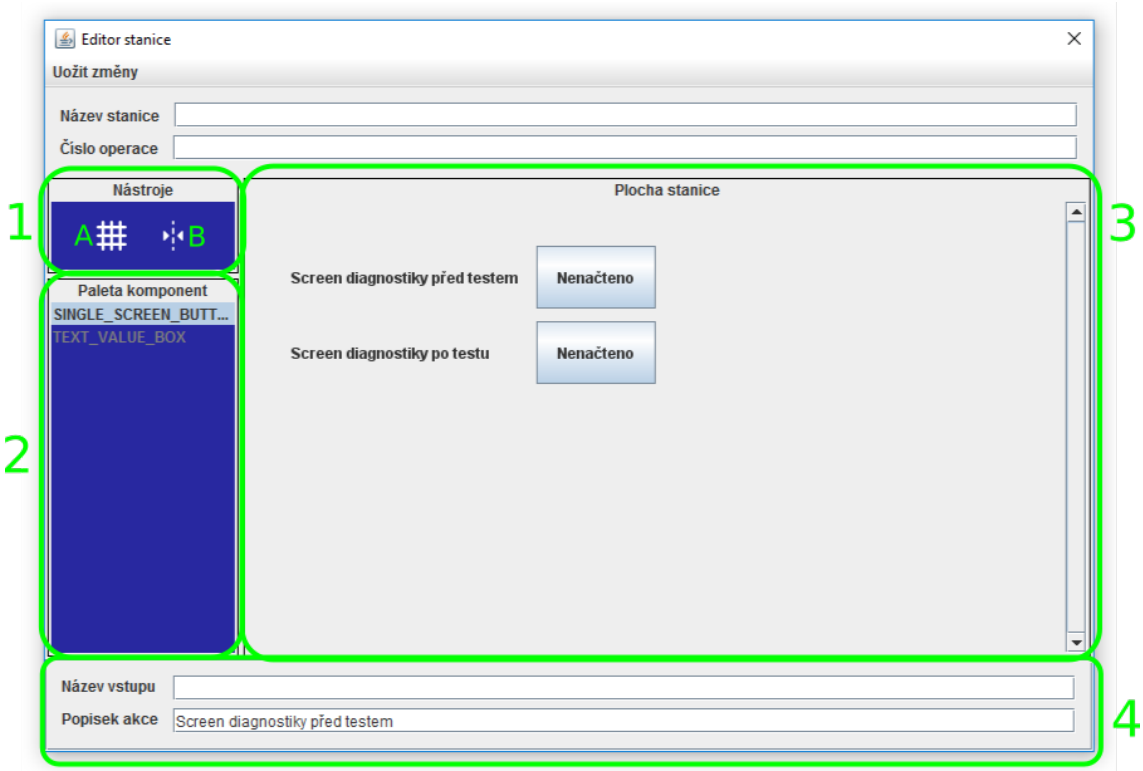
Aby vytváření panelů repasních stanic bylo uživatelsky přívětivé, intuitivní a zároveň umožňovalo pokročilé nastavení vzhledu přesně dle požadavků, byl implementován grafický editor podobný těm, které známe z vývojových prostředí pro návrh uživatelského rozhraní.

Editor umožňuje přetažením z palety komponent(10) vložit na plochu stanice nový prvek. Podrobnější popis jednotlivých prvků stanice naleznete v kapitole 6.4. Přidanou komponentu lze označit kliknutím myši, a poté ji tažením přesunout do nové pozice. Tažením v rozích komponenty lze libovolně modifikovat její velikost, musí však být zachována minimální velikost komponenty, která je dána minimální velikostí pro její správné zobrazení. Lze označit více komponent zároveň kliknutím na plochu editoru a tažením rámečku výběru přes požadované komponenty. Při uvolnění tlačítka myši jsou označeny všechny komponenty, jejichž průnik s rámečkem výběru není nulový. Dalším způsobem, jak vybrat více komponent je postupným výběrem za současného držení klávesy SHIFT. Všechny označené komponenty jsou při změně velikosti nebo přesunu jedné z nich modifikovány stejně. Označené komponenty lze smazat klávesou DELETE.

Při označení komponenty je ve spodní části editoru zobrazen panel pro editaci parametrů(viz bod 4 na obrázku 10). Krom identifikátoru komponenty umožňuje editovat další vlastnosti týkající se funkcionality nebo vzhledu. Více o možnostech nastavení pro jednotlivé komponenty naleznete v kapitole 6.4.

Jednotlivé komponenty stanice se nesmí překrývat. V případě, že při přesunu či transformaci komponent dojde k vzájemnému překrytí, jsou takto kolidující komponenty červeně zvýrazněny. Při ukončení akce jsou všechny překrývající se komponenty vráceny na své původní umístění.

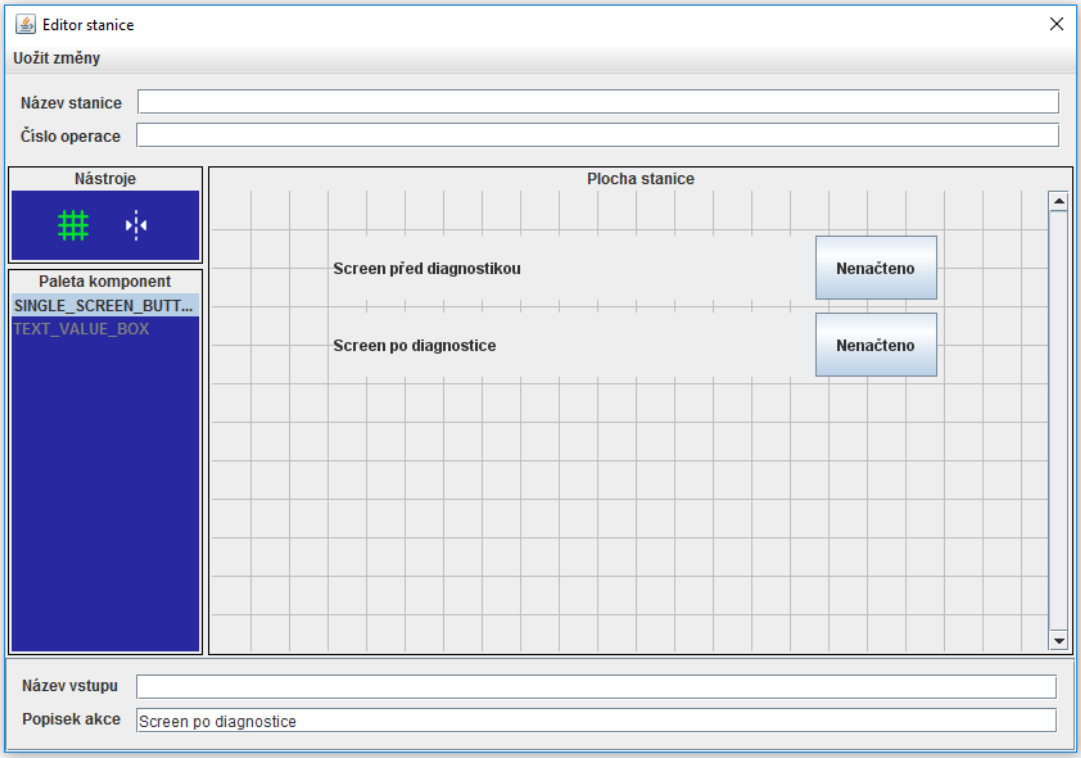
Při spuštění se editor nachází ve výchozím módu. V něm lze komponenty volně umisťovat a transformovat. Aby však výsledné uživatelské rozhraní bylo přehledné a



Obrázek 10: Grafický editor pro návrh stanice : (1) Panel nástrojů, (2) Paleta komponent, (3) Panel návrhu (náhled stanice), (4) Panel vlastností komponenty

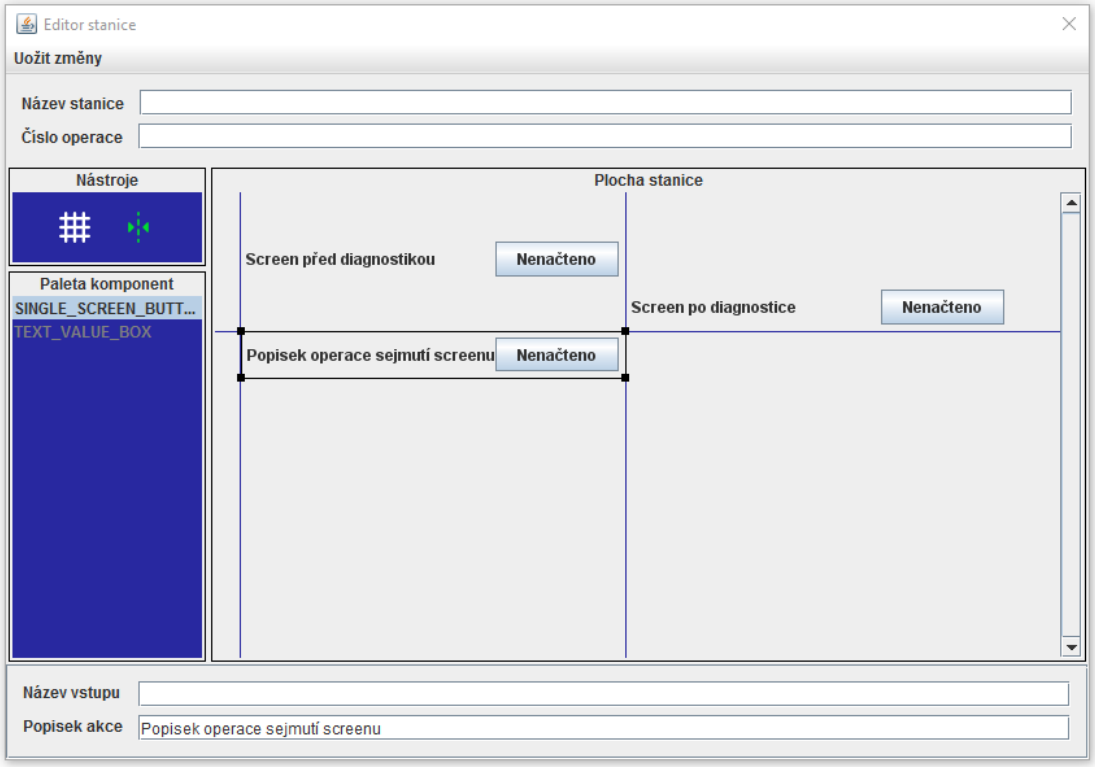
působilo konzistentně, je třeba jednotlivé komponenty vůči sobě zarovnat do bloku. V tomto případě by uživatel byl nucen každou komponentu zarovnat ručně, což by bylo časově náročné a frustrující. Proto jsou implementovány další dva módy, které uživateli zarovnání komponent usnadňují. Ty lze zapínat pomocí ikon v panelu „Nástroje“, viz 10, bod 1.

První mód lze zapnout ikonou znázorňující mřížku(10- 1A). V případě aktivace tohoto módu je na plochu designeru vykreslena mřížka, ke které jsou všechny přidané komponenty navázány. U komponent, které byly přidány před zapnutím tohoto módu, je upravena pozice a velikost tak, aby korespondovaly s vykreslenou mřížkou. Velikost buněk vykreslené mřížky je možné upravit kliknutím pravého tlačítka myši na ikonu mřížky a výběrem požadované velikosti. Tento mód umožňuje rychlý návrh stanice, neumožňuje však uživateli návrh přesně dle jeho představ.



Obrázek 11: Designer se zapnutou kotvící mřížkou

Druhý mód lze zapnout ikonkou zobrazující svislou přerušovanou čáru se dvěma trojúhelníky(10- 1B). Ten dovoluje libovolnou transformaci komponenty, obdobně jako výchozí mód editoru. Při přesunu nebo transformaci komponenty jsou však kontrolovány pozice zbylých komponent. V případě, že se krajní body modifikované komponenty nacházejí v dostatečně malé vzdálenosti od rovnoběžky s osou X (případně Y) procházející krajními body jiné komponenty, jsou posunuty na tuto osu. Při zarovnání komponenty je daná osa vykreslena.



Obrázek 12: Designer s aktivním zarovnáváním dle komponent

6.3 Implementace grafického editoru

Při návrhu struktury programu byly brány ohledy na budoucí rozšiřitelnost a přehlednost zdrojového kódu. Ten je díky tomu rozložen na mnoho vzájemně provázaných tříd implementujících různá rozhraní. Cílem následujících kapitol je usnadnit orientaci v této rozsáhlé struktuře a popsat základní nosné konstrukce programu.

Jednotlivé vstupy, které se na ploše stanice nacházejí, různými způsoby interagují s uživatelem a na základě těchto interakcí mění svůj vzhled (např. dovolují uživateli kliknutím spustit skript, vepsat text, označit zaškrtávací pole). Toto chování je však při použití v designeru nežádoucí. Aby nebylo nutné u každé komponenty implementovat možnost vypnutí těchto funkcí, byla vytvořena třída *StationEditorInputDecorator*. Jak již název napovídá, třída je z části postavena na návrhovém vzoru „Decorator“, který umožňuje rozšířit funkcionalitu objektu dynamicky za běhu aplikace.[12] Tato třída přebírá v konstruktoru instanci potomka třídy *A_StationInput*, která reprezentuje konkrétní komponentu a „dekoruje“ její

metodu *getGraphicsComponent()*. Implementaci samotných komponent popisuje kapitola č.6.5.

V rámci metody *getGraphicsComponent()* je nejprve zavolána stejná metoda nad objektem předaným v konstruktoru. Tím získáme objekt *JComponent* definující chování a vzhled komponenty vstupu stanice. Získanou komponentu lze dle potřeby modifikovat, tj. nastavit požadovanou velikost, předvyplnit zobrazená data apod. Následně je vytvořena nová instance třídy *JComponent*, která přetěžuje metodu *paint(Graphics)*, jenž se stará o vykreslení grafiky samotné. Přetížená metoda vytvoří grafický kontext z původní komponenty a ten poté vykreslí pouze jako obrázek. Výsledná komponenta tedy vypadá identicky, ale již nijak nereaguje na události kliknutí myší, stisku klávesnice apod.

Další výhodou tohoto řešení je již zmíněná možnost nastavení komponenty před vytvořením její grafické reprezentace. Při transformaci komponenty v editoru nebo aktualizaci jejích vlastností je tedy opakovaně generován grafický kontext. Díky tomu při zvětšování komponenty nedochází k poklesu kvality jako při transformaci rastrového obrázku, dochází k aktualizaci layout manažera komponenty a vlastnosti komponenty (např. popisky) jsou aktualizovány v reálném čase, takže uživatel v každém okamžiku vidí přesný vzhled.

6.4 Komponenty implementované v grafickém editoru

V rámci diplomové práce bylo vytvořeno několik základních komponent pro použití ve vytvořeném editoru. Ty lze rozdělit do dvou skupin: komponenty pro ukládání dat a pomocné komponenty pro řízení samotného procesu. Komponenty první skupiny umožňují ukládání nejčastějších druhů dat, se kterými se v průběhu repasování můžeme setkat. Pomocné komponenty naopak slouží k zobrazení důležitých informací z pracovního postupu. Seznam komponent byl sestaven na základě aktuálních požadavků na archivaci dat a řízení procesu repasování. Následuje výčet jednotlivých komponent s popisem jejich účelu i způsobu použití.

Hint Box

Komponenta pro zobrazení krátké textové nápovědy přímo na ploše stanice. Po přidání na plochu editoru má uživatel možnost vybrat ikonu zobrazenou vedle textu, na základě toho, zdali se jedná pouze o informaci, nebo důležitější varování. Také je zde možné nastavit tu část textu, která má být zobrazena vždy. V rámci nastavení vstupu v editoru reference pak lze zadat text, který je zobrazený pouze jako dodatek pro konkrétní referenci. V obou textových polích lze použít základní značky jazyka HTML pro formátování textu, vytváření seznamů apod. Tato skutečnost vychází z podpory HTML přímo v grafických komponentách knihovny Swing.[15] Oficiální dokumentace [15] bohužel neuvádí seznam podporovaných tagů, ani zdroj kde jej lze dohledat. Bližší informace lze nalézt v knize Core Swing: advanced programming[16]. Zde je uvedeno, že HTML tagy a atributy podporované HTML balíčkem, který je v knihovně Swing implementován, lze zjistit pomocí metod uvedených v ukázce 6.1. Na prvních dvou řádcích jsou uvedeny hlavičky zmíněných metod. Zbytek ukázky je kód použitý pro výpis podporovaných tagů a atributů na standardní výstup. Konkrétní výstup tohoto programu spuštěného na testovacím PC s nainstalovaným běhovým prostředím jazyka Java ve verzi 1.8.0_92 je následující:

Podporované tagy: a, address, applet, area, b, base, basefont, big, blockquote, body, br, caption, center, cite, code, dd, dfn, dir, div, dl, dt, em, font, form, frame, frameset, h1, h2, h3, h4, h5, h6, head, hr, html, i, img, input, isindex, kbd, li, link, map, menu, meta, nobr, noframes, object, ol, option, p, param, pre, samp, script, select, small, span, strike, s, strong, style, sub, sup, table, td, textarea, th, title, tr, tt, u, ul, var

Podporované atributy: face, comment, size, color, clear, background, bgcolor, text, link, vlink, alink, width, height, align, name, href, rel, rev, title, target, shape, coords, ismap, nohref, alt, id, src, hspace, vspace, usemap, lowsrc, codebase, code, archive, value, valuetype, type, class, style, lang, dir, declare, classid, data, codetype, standby, border, shapes, noshade, compact, start, action, method, enctype, checked, maxlength, multiple, selected, rows, cols, dummy, cellspacing, cellpadding, valign, halign, nowrap, rowspan, colspan, prompt, http-equiv, content, language, version, n, frameborder, marginwidth, marginheight, scrolling, noresize, media, endtag

Samozřejmě použití některých tagů je kontraproduktivní. Lze například vložit obrázek pomocí tagu <img>. Jako zdroj obrázku nesmí být použita webová adresa, lze použít pouze obrázky z lokálního zdroje, např: *<img src=„file:C:\zdroj.png“>*. Takto vložený obrázek však bude fungovat pouze na počítačích, na kterých bude na dané cestě obrázek uložen, jelikož tímto použitím samozřejmě nedojde k uložení zdrojového souboru do databáze.

```
1  // public static HTML.Attribute[] getAllAttributeKeys()
2  // public static HTML.Tag[] getAllTags()
3
4  import javax.swing.text.html.HTML;
5  import javax.swing.text.html.HTML.Attribute;
6  import javax.swing.text.html.HTML.Tag;
7  .
8  .
9  .
10 StringBuilder result = new StringBuilder();
11 for (Tag t : HTML.getAllTags()) {
12   result.append(t.toString()).append(", ");
13 }
14 result.setLength(result.length() - 2);
15 System.out.println("Podporované tagy: " + result.toString());
16
17 result.setLength(0);
18 for (Attribute a : HTML.getAllAttributeKeys()) {
19   result.append(a.toString()).append(", ");
20 }
21 result.setLength(result.length() - 2);
22 System.out.println("Podporované atributy: " + result.toString());
```

Ukázka zdrojového kódu 6.1: Výpis HTML tagů a atributů podporovaných knihovnou Swing

Help Button

Tlačítko určené zejména pro zobrazení pracovního postupu v aplikaci eDoc. Ta běží na webovém serveru v rámci interní sítě a obsahuje veškeré pracovní postupy, technické výkresy a další dokumentaci týkající se daného výrobku. Díky použití HTTP metody GET lze každý dokument zobrazit pomocí unikátní URL. Při přidání tlačítka v editoru stanice je jediným parametrem jméno. URL adresa cílového pracovního postupu je zadávána v rámci nastavení reference. Při stisknutí tlačítka

je otevřen výchozí webový prohlížeč se zadanou adresou.

Text Value Box

Komponenta určená pro zadávání textového řetězce. Aby byl návrh nové stanice co nejjednodušší, je součástí komponenty i popisek, není tedy nutné ho vkládat samostatně jako další komponentu. Může však zůstat prázdný, aby bylo možné s komponentou pracovat jako s klasickým textboxem známým z designérů uživatelských rozhraní. Krom jména vstupu a textu popisku lze v editoru stanice nastavit kontrolu formátu vkládaných dat výběrem jedné z následujících možností:

Alfanumerický kód (bez speciálních znaků) umožňuje zadat uživateli jednoslovný řetězec složený z číslic a znaků anglické abecedy. Je určen především pro vstupy určené k uchování čárových kódů načtených pomocí ruční čtečky (výrobní a sériová čísla).

Libovolný řetězec dovoluje uživateli zadat jakýkoliv text.

Celočíselná hodnota dovoluje vkládání pouze celých čísel. V nastavení stanice při vytváření konstrukčního typu nebo reference je u tohoto vstupu nutné vyplnit minimální a maximální přípustnou hodnotu. V případě, že poté v reportu uživatel zadá hodnotu mimo zadané meze, nelze již stanici dokončit jako úspěšnou (repasovaný díl nesplňuje specifikaci a nelze jej tedy úspěšně repasovat).

Neceločíselná hodnota je stejná jako filtr celočíselných hodnot zmíněný výše, s tím rozdílem, že je možné vkládat i reálná čísla s desetinou tečkou.

Regulární výraz dovoluje v nastavení stanice při vytváření reference (konstrukčního typu) specifikovat regulární výraz. Hodnoty zadané do daného vstupu stanice poté musí odpovídat danému výrazu. Tuto možnost lze využít například tehdy, když chceme zabránit použití komponenty z dané série (např. se sériovým číslem začínajícím písmeny AD apod.).

Item Select

Komponenta pro výběr jedné či více položek z definovaného seznamu. Počet položek v seznamu není omezen, komponenta je však určena spíše pro menší seznamy (tj. v řádu jednotek až desítek možností). Toto omezení je dané zejména stylem výběru položek. Pro každou položku v seznamu je vytvořeno zaškrtávací pole (check box) nebo přepínač (radio button) v závislosti na tom, zda uživatel při definici stanice povolil výběr jedné nebo více položek. Při velkém počtu možností by vyhledání konkrétní položky bylo zdlouhavé.

Při vložení komponenty na plochu stanice je nutné vyplnit název komponenty, popisek zobrazený jako součást rámečku komponenty, volbu zdali lze vybrat pouze jednu či více možností a název seznamu možnosti výběru. Tento seznam obsahuje obvykle velké množství možností, proto se konkrétní položky, ze kterých bude možné vybírat specifikují až v nastavení stanice pro konkrétní referenci.

Příkladem důvodu k přesunutí definice seznamu možností z editoru stanice do editoru reference může být použití pro výběr vyměněných komponent při repasi výkonového obvodu, který se stará o buzení motoru EPHS pumpy. Výkonové obvody jsou u různých pump principiálně podobné a skládají se z omezeného počtu elektronických součástek, které lze v procesu měnit (budící tranzistory, tlumivky, kondenzátory apod.). Konkrétní řešení pro různé výrobky se ale mírně liší. Např. výkonový obvod pumpy vozidla BMW mini obsahuje pouze dva budící tranzistory díky použití stejnosměrného komutátorového motoru. Velká část moderních hydraulických čerpadel posilovačů řízení (z vozidel Škoda Fabia, Opel Astra H atd.) však využívá stejnosměrné BLDC motory obsahující obvykle 3 vinutí, kde každé je spínáno jedním tranzistorem. Lze tak pro každou referenci přesně nastavit seznam komponent tak, aby odpovídal konstrukci dané pumpy a v seznamu nebyly nadbytečné položky.

CMD Run

Jak již název komponenty napovídá, je určena k provádění příkazů přes příkazovou řádku operačního systému. Lze ji využít ke spouštění testovacích programů (např. programu FDT, více viz kapitola 3), stejně jako k načítání textových souborů vy-

tvořených aplikacemi třetích stran apod.

Po přidání komponenty na plochu stanice je nutné nastavit pouze jméno komponenty. Textový popisek, zobrazený vlevo od tlačítka pro spuštění příkazu, je volitelný. Pokud jej tedy uživatel ponechá prázdný, tlačítko zabírá celou plochu komponenty. Příkaz, který má být proveden po stisknutí tlačítka, je nastaven až při vytváření reference, aby jej bylo možné upravit přesně podle potřeb (např. z důvodu konkrétního nastavení testovacího programu).

Text Viewer

Jedná se o komponentu úzce spojenou s komponentou *CMD Run*. Zastupuje skupinu komponent, které jsou určeny k zobrazení výstupu provedeného příkazu.

Na ploše stanice je reprezentována textovým polem, do kterého je přesměrován standardní výstup z příkazové řádky. Při jejím použití je nutné nastavit jméno komponenty, popisek zobrazený v rámečku a komponentu typu *CMD Run*, ze které je získán výstup. Díky tomu lze k jedné komponentě typu *CMD Run* připojit více komponent určených k zobrazení a zpracování dat tak, aby mohl být výstup reprezentován různými způsoby přesně dle požadavku uživatele (po vzoru této komponenty je možné implementovat další, které budou textový výstup např. parsovat a získané hodnoty vykreslovat v podobě grafu).

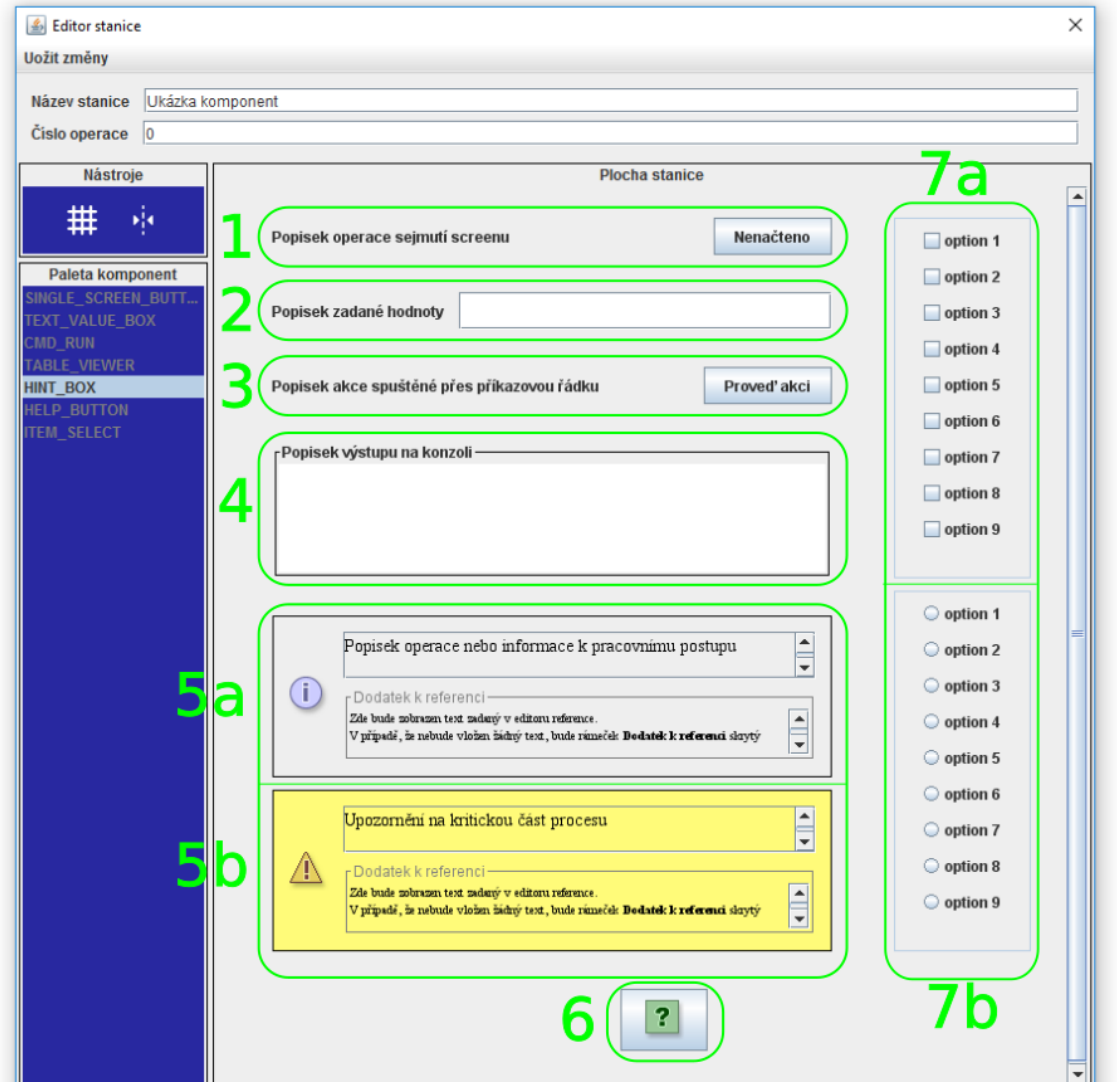
Každá komponenta pro zobrazení výstupu předaného komponentou *CMD Run* musí implementovat rozhraní *I_InputStreamViewer* obsahující následující metody:

void processStarted()	volána ihned po stisknutí tlačítka pro spuštění procesu
void appendLine(String line)	volána po každém vypsání nové řádky na standardní výstup
void appendError(String error)	volána po každém vypsání nové řádky na chybový výstup
void processFinished()	volána po dokončení procesu a uzavření výstupních proudů

void processStopped(Exception e)	vyvolána v případě, že došlo k výjimce při vstupně/výstupní operaci (ukončení proudu dat apod.)

Single Screen Button

V některých případech jsou v procesu repasování využívány programy třetích stran, které neposkytují žádnou možnost získání dat vyjma grafického výstupu v podobě vykreslení grafů nebo vyplnění hodnot do formuláře samotné aplikace. Pro tyto případy byla implementována komponenta *Single Screen Button*, která uživateli umožňuje rychle a snadno vytvořit snímek obrazovky, neboli tzv. „screenshot“. Po vložení na plochu stanice je nutné nastavit název komponenty a popisek zobrazený vedle tlačítka pro vytvoření snímku, který uživateli popisuje co by mělo být obsahem uloženého obrázku. V rámci nastavení stanice pro konkrétní referenci je možné specifikovat, zdali má být vytvořen snímek celé obrazovky nebo pouze výřezu vybraného uživatelem. V prvním případě je screen pořízen pouhým stisknutím tlačítka. Opětovným stiskem lze zobrazit okno s náhledem a volbou umožňující pořízený obrázek smazat. Pokud má být pořízen pouze výřez obrazovky, jsou po stisku tlačítka skryta všechna okna aplikace a je uživateli umožněno tažením rámečku výběru pomocí myši vybrat požadovanou oblast.



Obrázek 13: Grafická ukázka všech komponent, které lze využít při návrhu grafického rozhraní repasovací stanice

1 Single Screen Button

2 Text Value Box

3 CMD Run

4 Text Viewer

5 Hint Box nastavený pro zobrazení:
a) informace b) varování

6 Help Button

7 Item select nastavený pro:
a) výběr více možností
b) výběr jedné z možností

6.5 Implementace nových komponent

Přestože bylo vytvořeno několik základních komponent, je pravděpodobné, že s postupem času vyvstane potřeba implementovat novou komponentu splňující aktuální požadavky. Následující kapitola popisuje hierarchii tříd použitých při návrhu komponent a slouží jako návod k vytvoření nové komponenty.

Dosud vytvořené komponenty lze nalézt v balíčku *remsys.gui.station.inputs*. Zde je pro každou z komponent vytvořen balíček nesoucí její název. Název komponenty, který je použitý jako popisek na paletě komponent, musí být také přidaný do výčtu *E_InputType*. Ve zmíněném balíčku se současně nacházejí abstraktní třídy sloužící jako předci pro jednotlivé části tvořící výslednou komponentu: grafický objekt reprezentující onu komponentu, panel pro nastavení základních vlastností komponenty a panel nastavení pro specifikaci chování komponenty pro danou referenci.

Každá komponenta, která může být přidána na plochu stanice, musí rozšiřovat třídu *A_StationInput*. Ta obsahuje objekt typu *StationInputData* uchovávající informace o umístění a velikosti komponenty na ploše stanice, případně i objekty nesoucí informace o reportu a stanici, na které se komponenta nachází. Které objekty komponenta obsahuje je dáno použitým konstruktorem. První slouží k vytvoření komponenty pro použití v editoru stanice, druhý pro vytvoření komponenty přímo pro stanici načtenou pro konkrétní report.

Dále třída *A_StationInput* implementuje následující rozhraní :

- *I_GraphicsComponent* - Rozhraní pro třídy, které tvoří součást GUI (tzn. že jsou potomkem třídy JComponent z knihovny Swing, nebo takového potomka obsahují v atributech třídy). Obsahuje metodu pro získání grafické reprezentace komponenty implementované pomocí objektů knihovny Swing.

- *I_ChangesTestable* - Rozhraní obsahující metody pro uchování aktuálního stavu třídy a kontrolu, zdali se daný stav od posledního uložení změnil. Třída implementující toto rozhraní musí sama obsahovat atributy pro uložení aktuálního stavu. Toto rozhraní umožňuje uživateli zobrazit dotaz na uložení provedených změn v případě, kdy byla např. změněna hodnota komponenty při její editaci.

- *I_ValidityDisplayable* - Toto rozhraní je určeno zejména pro komponenty GUI, které mohou uživatele informovat o zadání nevalidních dat změnou svého vzhledu, např. červeným zabarvením apod.

- *I_RelevanceDisplayable* - Rozhraní pro komponenty stanice. Obsahuje metody pro nastavení a načtení důležitosti vyplnění.

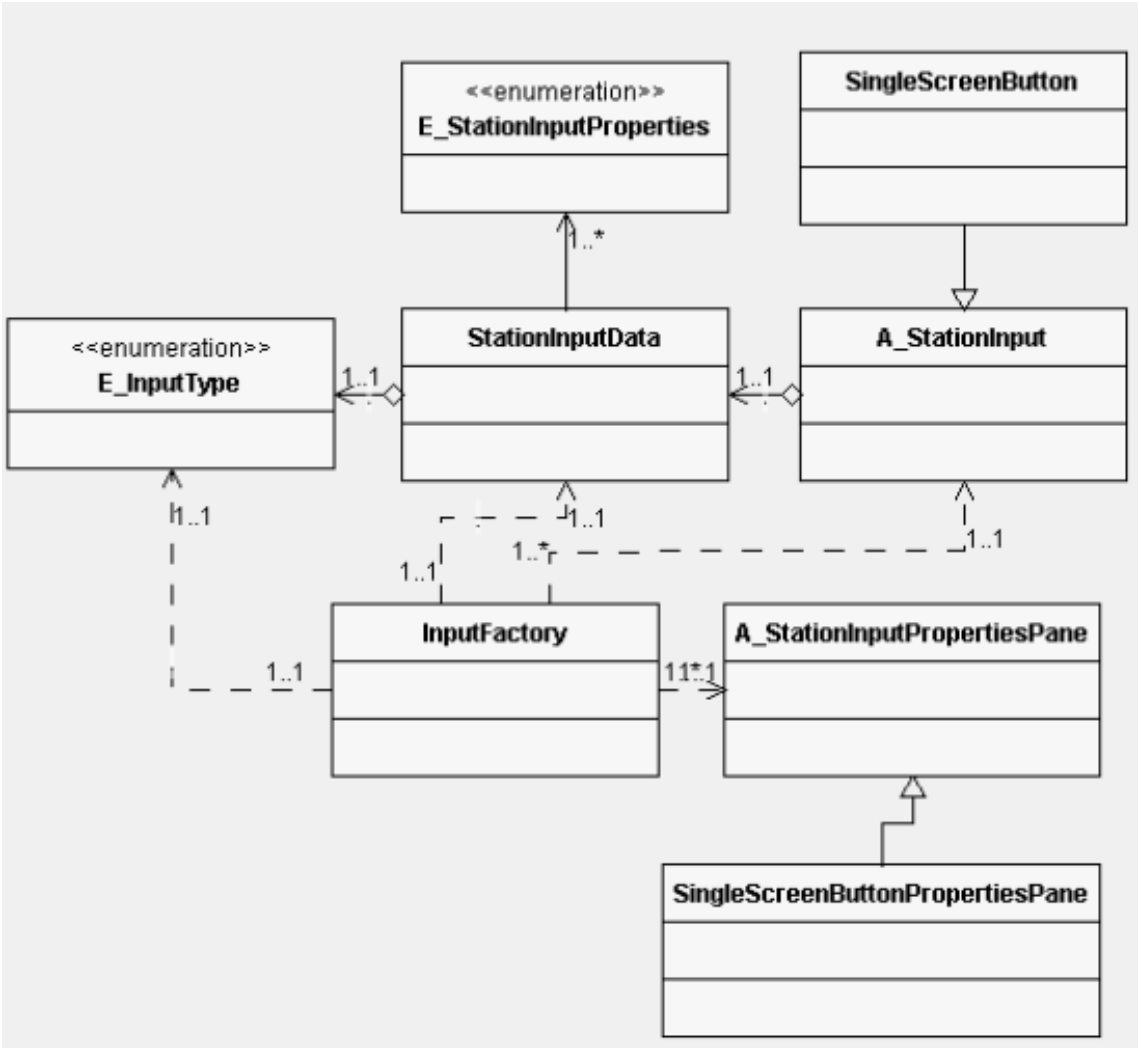
- *I_Editable* - Rozhraní obsahující metody pro nastavení možnosti editace. Komponenty na stanici, která je zobrazena pouze pro čtení nesmějí být editovatelné. Toho lze docílit i voláním metody *setEnabled(Boolean enabled)*, kterou implementuje třída JComponent z knihovny Swing. Některé komponenty knihovny Swing však při volání této metody mění svůj vzhled nebo chování, což může negativně ovlivnit požadovaný výsledek (např. textové pole při zakázání editace změní barvu pozadí a znemožní označit vložený text, což zároveň zhorší čitelnost vloženého textu a nedovolí uživateli vložený text označit pro kopírování). Proto je pro nastavení editovatelnosti použita metoda tohoto rozhraní, ve které lze upravit vzhled a chování komponenty dle potřeby.

V neposlední řadě obsahuje abstraktní metody pro načítání/ukládání dat z/do databáze a pro nastavení/získání dalších vlastností vstupu.

Samotnou funkcionalitu a definici vzhledu komponenty již obsahují konkrétní potomci této třídy. Pro získání informací o ostatních komponentách nebo navěšení obsluhy událostí stanice můžeme využít metodu *getParentStation()*. V případě, že implementujeme komponentu jejímž účelem není ukládat získaná data, ale slouží pouze jako informační prvek, můžeme implementovat metody pro načítání a ukládání dat z databáze pouze jako prázdné metody.

Na obrázku 14 jsou třídy konkrétních komponent zastoupeny třídou *SingleScreenButton*.

Zároveň musí být pro každou komponentu definován panel pro nastavení vlastností komponenty, který je potomkem abstraktní třídy *A_StationInputPropertiesPanel*. Tato třída rozšiřuje třídu JPanel z knihovny Swing a je zobrazena v editoru stanice při označení příslušné komponenty (viz bod 4 na obrázku 10). Při implementaci nového panelu musí být implementovány tři abstraktní metody předka: metoda pro kontrolu validity dat zadaných uživatelem, metoda pro získání vlastností a je-



Obrázek 14: Diagram tříd: komponenty stanice

jich hodnot a metoda pro zpětné vyplnění panelu pomocí vlastností načtených z databáze. Dále je samozřejmě nutné v konstruktoru nadefinovat grafický vzhled panelu se všemi ovládacími prvky. Pokud jedna z vlastností přímo ovlivňuje vzhled komponenty, pak lze vyvolat obnovení grafiky komponenty s aktuálně vyplněnými údaji voláním metody *getParentStationComponent().updateCanvas()*. Mezi takové vlastnosti řadíme např. popisek komponenty zobrazený uživateli. V tomto konkrétním případě bychom volání výše uvedené metody přidali do obsluhy události změny obsahu textového pole (viz ukázka 6.2)

Poslední třídou, kterou je nutné pro každou komponentu vytvořit je potomek třídy *A_StationInputSettingsPanel*. V konstruktoru je třeba zavolat metodu *initLayout()*.

```
1  JTextField caption = new JTextField();
2  ...
3  caption.getDocument().addDocumentListener(new DocumentListener() {
4    @Override
5    public void insertUpdate(DocumentEvent e) {
6       getParentStationComponent().updateCanvas();
7    }
8
9    @Override
10   public void removeUpdate(DocumentEvent e) {
11           getParentStationComponent().updateCanvas();
12   }
13
14   @Override
15   public void changedUpdate(DocumentEvent e) {
16           getParentStationComponent().updateCanvas();
17   }
18 });
```

Ukázka zdrojového kódu 6.2: Použití metody *updateCanvas()*

Pro vytváření instancí objektů komponent a panelů nastavení slouží třída *InputsFactory*, která byla navržena dle návrhového vzoru Factory Method. Díky tomu by měl být minimalizován zásah do programu v případě implementace nového typu vstupu stanice[11]. Umožňuje vytvářet komponenty stanice na základě objektu *StationInputData* sestaveného z hodnot načtených z databáze. Obsahuje také metodu pro vytvoření prototypu komponenty (komponenty s výchozím nastavením vlastností a velikosti), nebo panelu nastavení pouze na základě typu daného výčtem *E_InputType*. Ty jsou využité při přidávání komponent z palety. V takovém případě je komponenta nastavena pomocí definovaných hodnot (výchozí jméno komponenty, výchozí popisek apod.).

Po implementaci nové komponenty je nutné nechat program vytvořit novou databázi, nebo ručně zadat hodnoty týkající se nové komponenty do příslušných tabulek. Veškeré výčtové typy jsou totiž při vytvoření databáze nakopírovány do tabulek, aby bylo možné při práci s databází spojit číselné identifikátory vlastností a názvů komponent s jejich názvem a zvýšit tak čitelnost dat při dotazování přímo na databázi.

7 Implementace ukázkové linky

Způsob použití nové aplikace je prezentován na vytvoření linky ECU a reference obsahující nastavení procesu repase řídící jednotky pro hydraulickou pumpu vozu BMW Mini.

Po spuštění aplikace je nutné se přihlásit uživatelským jménem a heslem. Pro vytváření stanic je nutné, aby přihlášený uživatel měl oprávnění úrovně „Technik“. Při úspěšném přihlášení je zobrazeno okno s výběrem základních operací: založení nové zakázky, načtení již existující zakázky a vyhledáním komponenty pomocí unikátního identifikátoru. V případě, že uživatel nechce provést žádnou z těchto operací, může zvolit čtvrtou možnost - zobrazení hlavního dialogového okna programu.

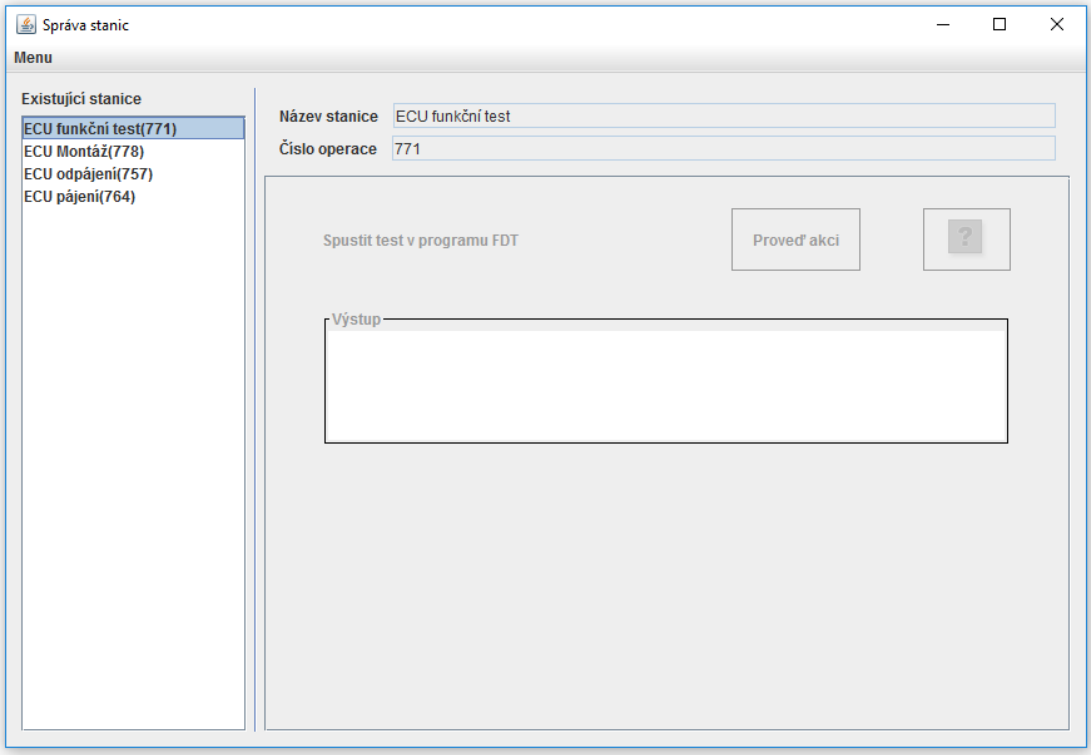
Založení nové stanice do systému lze provést výběrem možnosti *Správa stanic* v menu *Správa objektů* v hlavním dialogovém okně programu. Po výběru dané možnosti je zobrazen formulář uvedený na obrázku 15.

Ten obsahuje seznam všech dosud založených stanic. Při označení stanice v seznamu je vykreslen náhled stanice a její základní údaje. Výběrem položky v menu lze založit novou stanici a editovat či odstranit vybranou stanici. Tím se však nijak nenaruší již založené konstrukční typy či reference. Změny se projeví pouze při založení nového konstrukčního typu. Při založení stanice je zobrazen editor popsaný v kapitole 6.2.

Linka ECU obsahuje celkem šest stanic (v závorce je uvedeno unikátní číslo operace) [17]:

1. ECU mytí (750)
2. ECU odpájecí stanice (757)
3. ECU pájecí stanice (764)
4. ECU funkční test (771)
5. ECU montáž (778)
6. ECU balení a vysoušení (781)

Na stanicích 1. a 6. není umístěn počítač, proto nejsou do v programu založeny.



Obrázek 15: Formulář správy stanic: vlevo - seznam všech založených stanic vpravo - náhled a detaily právě vybrané stanice

Pro zbylé stanice jsou v souladu s pracovním postupem v programu vytvořeny jejich grafická rozhraní (bližší popis komponent naleznete v kapitole 6.4):

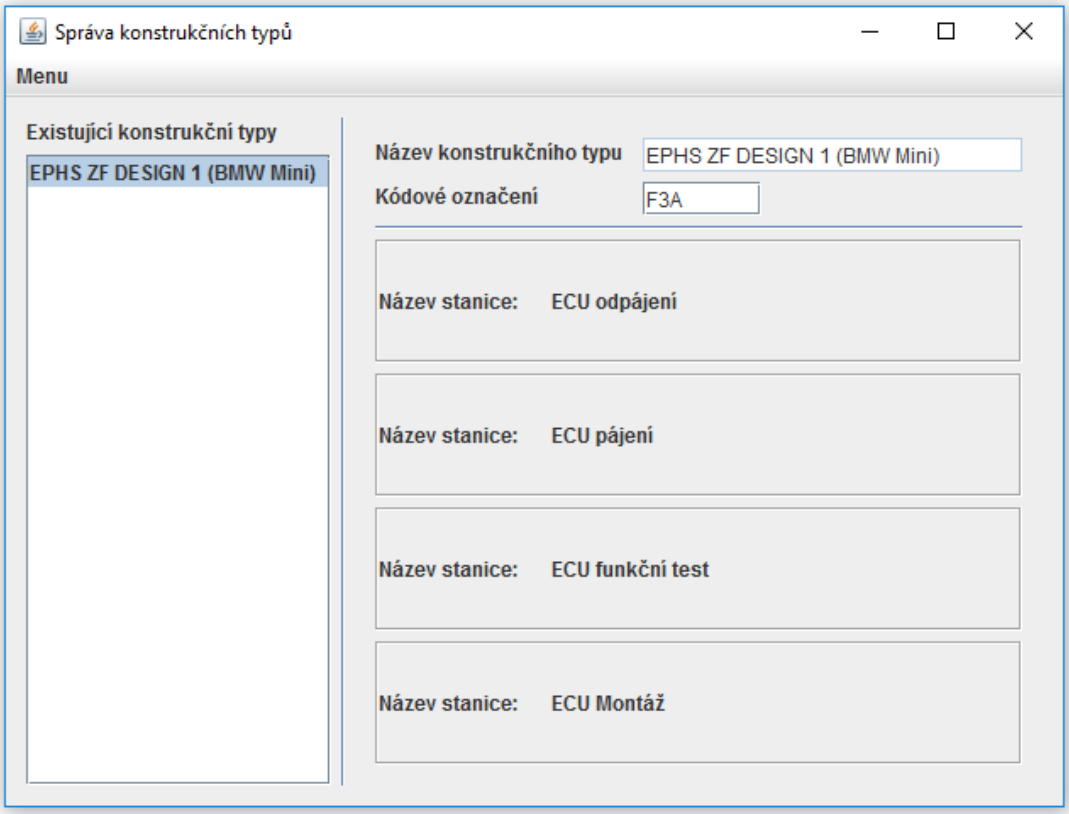
ECU odpájecí stanice obsahuje komponentu *Help Button* pro zobrazení aktuálního pracovního postupu a komponentu *Item Select* pro výběr odpájených komponent

ECU pájecí stanice obsahuje komponenty *Help Button* pro zobrazení aktuálního pracovního postupu, *Item Select* pro výběr připájených komponent a *Hint Box* zobrazující upozornění na zpětnou kontrolu kvality připájení komponent

ECU funkční test obsahuje komponentu *Help Button* pro zobrazení aktuálního pracovního postupu, komponentu *CMD Run* pro spuštění testování pomocí programu FDT přes příkazovou řádku a komponentu *Text Viewer* pro zobrazení výsledku tohoto testu.

ECU montáž obsahuje komponentu *Help Button* pro zobrazení aktuálního pracovního postupu

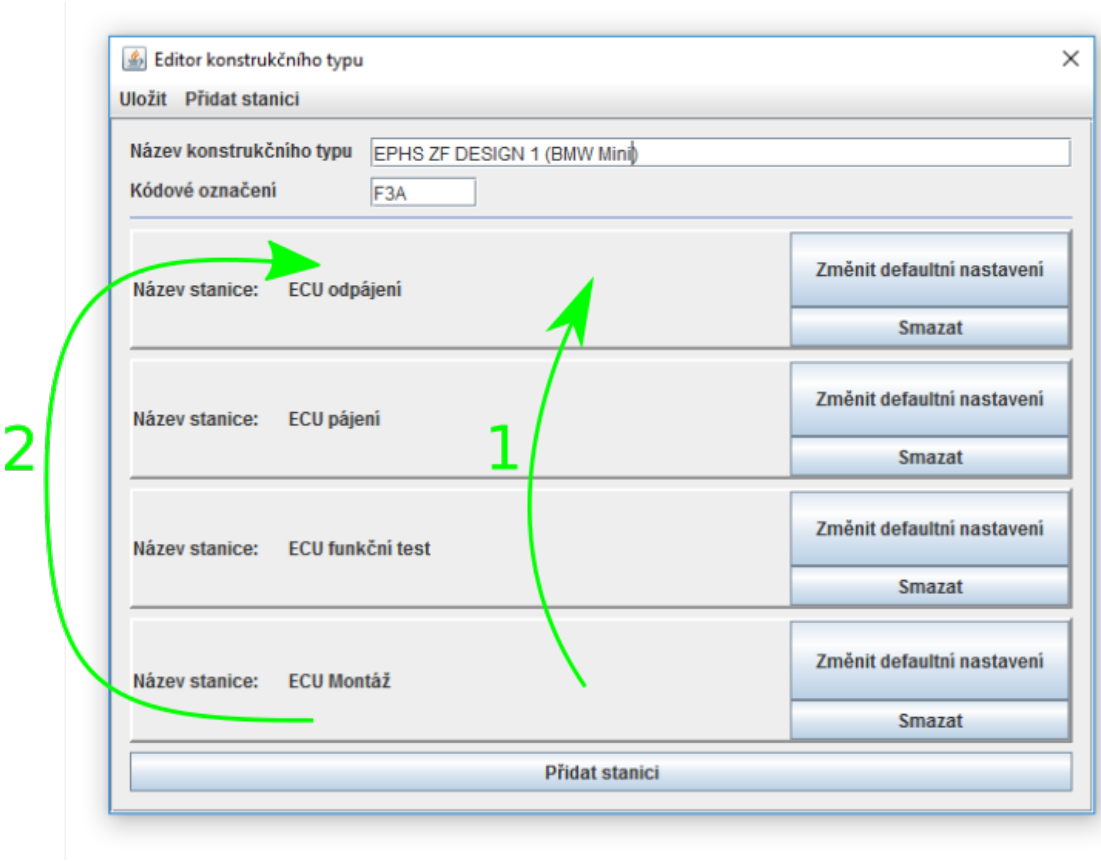
Po vytvoření jednotlivých stanic je nutné nadefinovat nový konstrukční typ. Seznam vytvořených konstrukčních typů a možnosti jejich vytváření a editace jsou přístupny z formuláře pro správu konstrukčních typů, jehož náhled vidíme na obrázku 16. Ten je, stejně jako správce stanic, dostupný z menu *Správa objektů* pod položkou *Správa konstrukčních typů*. Rozmístění prvků na formuláři je velmi podobné správci stanic z důvodu usnadnění orientace a ovládání programu.



Obrázek 16: Formulář správy konstrukčních typů:
vlevo - seznam všech založených konstrukčních typů
vpravo - náhled a detaily právě vybraného konstrukčního typu

Dle matice kódů konstrukčních typů [5] se hydraulická čerpadla posilovačů řízení, vyrobená firmou ZF Lenksysteme, dělí na dvě designové řešení. Naše zmiňovaná pumpa využívá designu 1. Odtud pramení název a kódové označení konstrukčního typu. Na obrázku 17 je zobrazeno okno editoru konstrukčního typu. Krom textových vstupů pro název a kódové označení můžeme na prázdném formuláři najít

pouze tlačítko *Přidat stanici*. Stiskem tlačítka je zobrazen dialog s výběrovým polem obsahujícím seznam dostupných stanic. Po zvolení stanice je na plochu editoru přidán panel reprezentující vybranou stanici, jehož součástí je její název a tlačítka pro změnu výchozího nastavení a pro opětovné odebrání stanice. Je povoleno přidat vícekrát stejnou stanici.

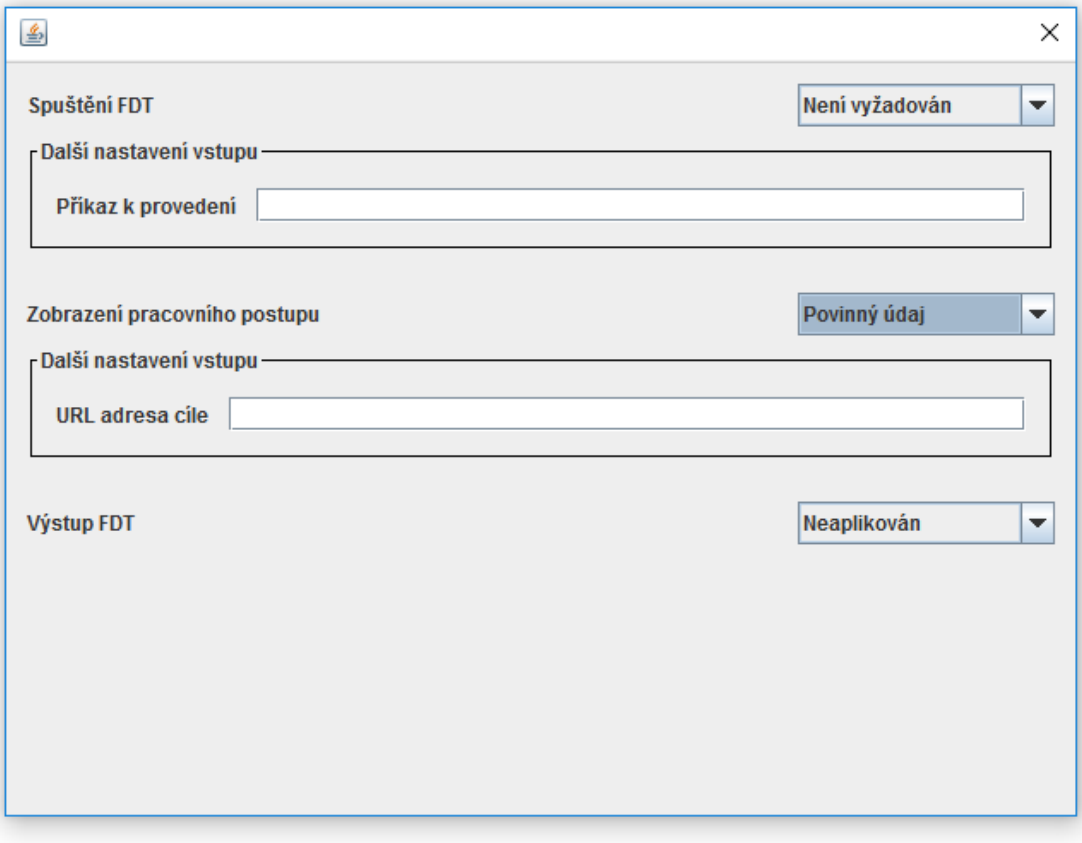


Obrázek 17: Designer konstrukčních typů - přesun poslední stanice na první pozici. Šipky naznačují trasu kurzoru za současného držení myši.
1) Výsledné pořadí: montáž, odpájení, pájení, funkční test
2) Výsledné pořadí: montáž, pájení, funkční test, odpájení

Stanice není potřeba přidávat v požadovaném pořadí, lze je libovolně přesouvat pomocí kliknutí na daný panel a tažení myší. Přesunem přes jiný panel se tento panel posune na aktuálně volnou pozici. Například při přesunu panelu, který je v seznamu poslední, přes všechny ostatní panely až na první místo v seznamu, dojde k posunu všech panelů o jednu pozici dolů a nikoliv k záměně prvního a posledního panelu.

Pokud však potřebujeme pouze zaměnit dva panely, je nutné vyhnout se přejezdu kurzoru myši přes ostatní panely. Po uchopení panelu stanice stiskem levého tlačítka myši je nutné vyjet kurzorem myši z dialogu bez kontaktu kurzoru s kterýmkoliv jiným panelem, viz obrázek 17.

Pro každou stanici lze stisknutím tlačítka *Změnit defaultní nastavení* vyvolat dialog pro nastavení výchozích hodnot jednotlivých vstupů stanice. Tyto hodnoty poté budou předvyplněny při vytváření nové reference. Dialog je generován dynamicky na základě jednotlivých vstupů stanice. Příklad dialogu pro stanici funkčního testu z ukázkového konstrukčního typu je na obrázku 18.



Obrázek 18: Dialog pro definici výchozího nastavení konstrukčního typu

Každý vstup je zde zastoupen svým jménem zadaným v editoru stanice a výběrovým polem určujícím prioritu vstupu.

Priorita vyplnění nabývá jedné ze tří možných hodnot:

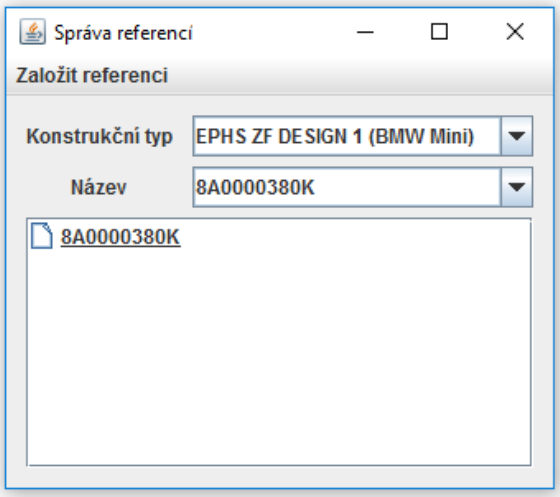
Neaplikován - vstup není aktivní

Není vyžadován - vstup je aktivní, ale není nutné jej vyplnit pro úspěšné dokončení stanice

Povinný údaj - vstup je zvýrazněn žlutou barvou a je nutné jej vyplnit (případně spustit v případě komponenty CMD Run) před dokončením stanice a jejím uzavřením v programu

Po otevření dialogu jsou všechny vstupy nastaveny na volbu *Neaplikován*. Po změně priority na jednu ze zbylých hodnot je pod řádkem reprezentujícím daný vstup zobrazen panel obsahující doplňující nastavení konkrétního vstupu, například se vstupem pro URL pracovního postupu apod. Jelikož jde pouze o nastavení výchozích hodnot, není zde kontrolována validita vstupů. Samotná kontrola je prováděna až v nastavení reference.

Po založení konstrukčního typu následuje založení reference. Seznam existujících referencí lze prohlížet pomocí formuláře *Správce referencí* (obrázek 19), který je dostupný z menu *Správa objektů* v hlavním okně aplikace. Zde lze zakládat nebo editovat reference všech existujících konstrukčních typů. Vybráním již vytvořené reference ve výběrovém poli *Název* je vypsán strom navazujících referencí.

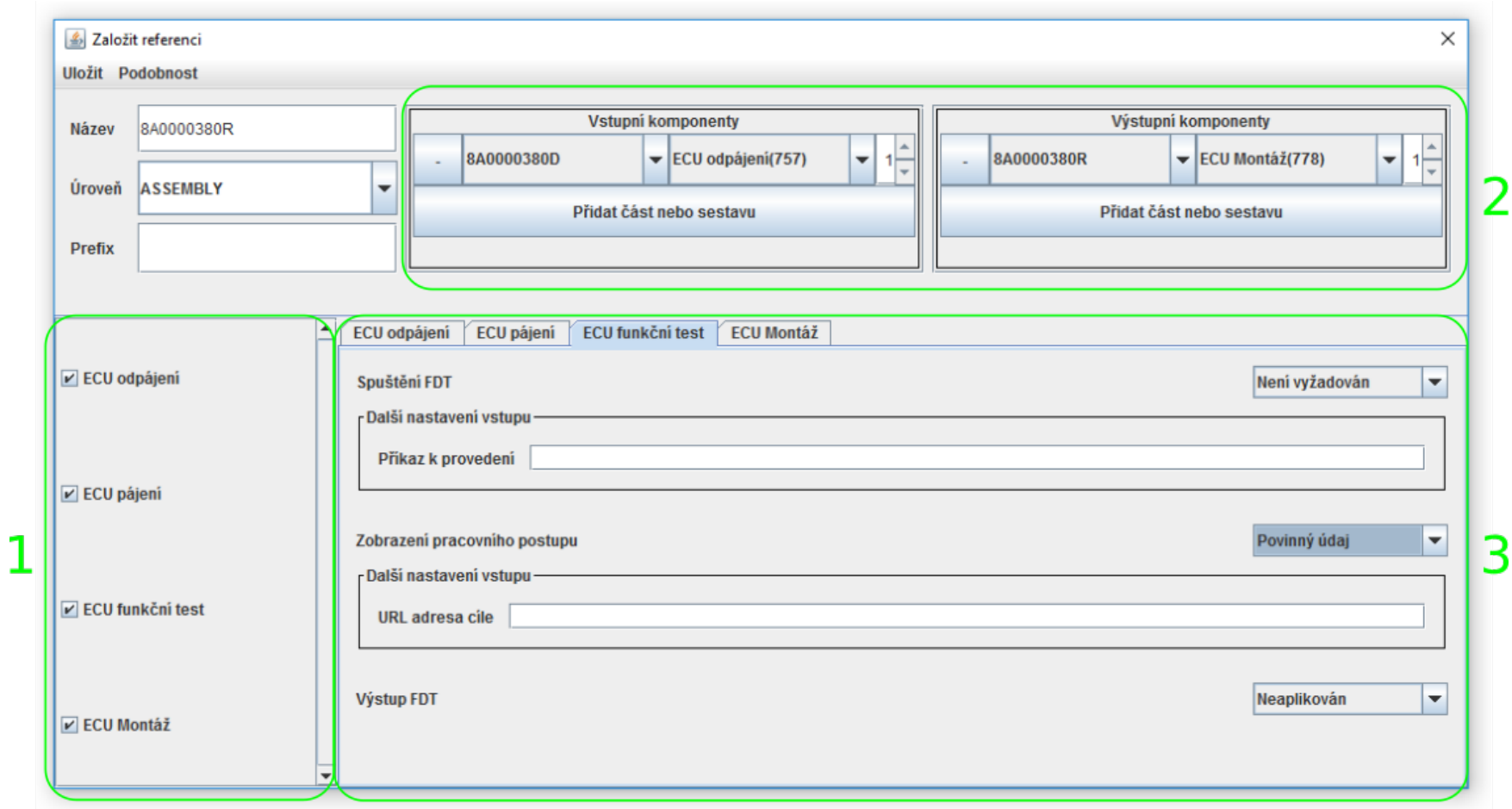


Obrázek 19: Formulář pro správu referencí

Navazující reference jsou ty reference, jejichž vstupní či výstupní komponenty odpovídají komponentám vybrané reference. V našem konkrétním případě jde tedy o referenci, v rámci které je z původní pumpy demontováno ECU jež vstupuje do právě vytvářené reference. Ve vykresleném stromu by šlo o položku nadřazenou. Současně by zde byla reference, v rámci které se montuje ECU a motor pumpy do jednoho celku, zobrazená jako podřízená položka.

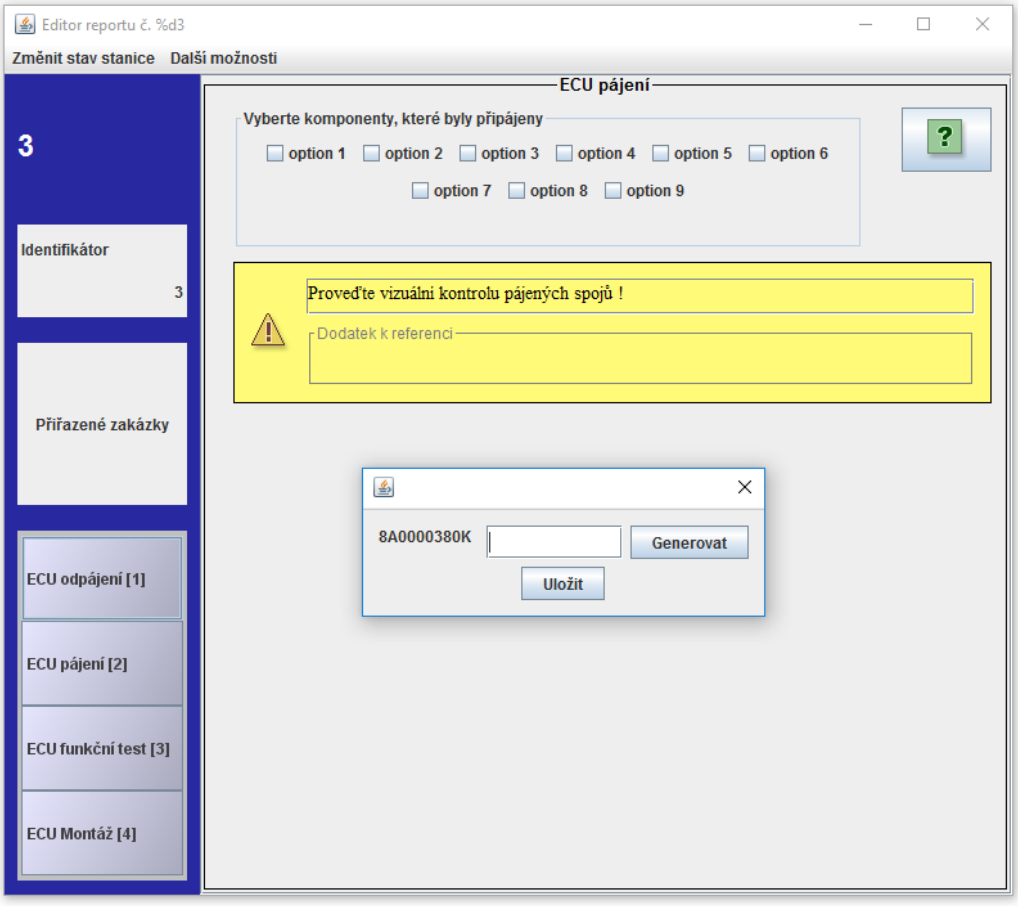
Při přidání nové reference se uživateli zobrazí editor reference vyobrazený na obrázku 20. V první řadě je zadán název reference a ze seznamu je vybrán typ zakázky. Ten má spíše informační charakter, umožňuje ale zároveň lepší kontrolu správnosti nastavení komponent. Komponenty, které vstupují nebo vystupují z procesu jsou zadávány pomocí panelů v oblasti 2 na obrázku 20. Stisknutím tlačítka *Přidat část nebo sestavu* uživatel přidá do téže panelu řádek pro nastavení komponenty. Tlačítko - umožňuje řádek opět odebrat. Další v pořadí je výběrové pole se všemi komponentami v databázi. Následuje výběrové pole pro výběr stanice, na které komponenta do procesu vstupuje nebo z něj vystupuje. Obsah pole je závislý na výběru stanic v oblasti 1 na obrázku 20. Na tomto panelu jsou vypsány všechny stanice, které byly přidány při definování konstrukčního typu. Z důvodů uvedených v kapitole 2 je zde možné vybrat pouze stanice, kterými výrobek dané reference skutečně prochází. Ve výběrovém poli stanic při nastavování komponent jsou zobrazeny pouze vybrané stanice.

Výběrem stanice v oblasti č.1 je současně přidána záložka se jménem stanice do oblasti 3 (viz obrázek 20). Každá záložka obsahuje možnosti pro nastavení vstupů, stejně jako při editaci výchozího nastavení konstrukčního typu na obrázku 18. Panel je přednastaven tak, jak byl nastaven při editaci konstrukčního typu, lze jej však přenastavit přesně dle potřeb.



Obrázek 20: Formulář editace reference: 1) volba použitých stanic, 2) nastavení komponent, 3) panel nastavení vstupů

Tím je příprava pro použití při repasování hotova. Následuje založení zakázky do systému. Po přidání nového kusu do zakázky se uživateli zobrazí dialog pro zadávání hodnot získaných během procesu (viz obrázek 21). V levé části dialogu jsou tlačítka pro přepínání mezi stanicemi. Pokud uživatel nemá oprávnění k zobrazení všech stanic, je na jednom pracovišti editovatelná pouze příslušná stanice. Při zobrazení stanice, na které do procesu vstupuje komponenta, je vygenerován dialog, který vidíme v popředí. Zde je prostor pro zadání čárového kódu nebo jiného identifikátoru umístěného na dané komponentě. Pokud žádný takový identifikátor není, lze tlačítkem *Generovat* vygenerovat kód složený z číslic a písmen anglické abecedy. Kód odpovídá internímu identifikátoru komponenty použitému v databázi, s tím, že je dané číslo převedeno do číselné soustavy o základu 36 (znaky 0-9A-Z) Stejný dialog se v případě výstupních komponent zobrazuje při ukončení prací na dané stanici.



Obrázek 21: Formulář pro zadávání hodnot získaných během procesu

8 Závěr

V rámci diplomové práce byla stávající aplikace, vytvořená pro repasní linku elektricky posilovaných sloupků řízení, modifikována tak, aby mohla být použita na linkách využívajících jiný způsob skladového hospodářství. Původní struktura databáze, navržená pro systém uchování tzv. K-kitů, byla přepracována tak, aby umožňovala uchovávat informace o jednotlivých komponentách a udržovala mezi nimi vazby ve skladovacím systému, ve kterém je repasovaný díl uskladněn rozebraný na jednotlivé komponenty. Z důvodu využití tohoto systému na více linkách – včetně plánované změny systému i na lince elektrických sloupků řízení – byla do databáze nově implementována rozsáhlá struktura umožňující uchovávat nastavení pro více repasních linek. To je zásadní rozdíl oproti původní databázi, která byla vytvořena pouze pro uchování dat na jedné konkrétní lince.

Současně s tím byla upravena samotná aplikace po vzoru vytvořené databáze. V aplikaci byla nově implementována možnost spravovat a vytvářet jednotlivé stanice, které jsou analogií k fyzickým stanicím umístěným na repasních linkách a které byly v původní aplikaci naprogramovány přímo ve zdrojovém kódu. Pro vytváření nových stanic byl vytvořen WYSIWYG grafický editor pro návrh uživatelského rozhraní, který umožňuje uživateli rychle a efektivně umístit na stanici vstupy pro data získaná z procesu repasování, stejně jako pomocné komponenty nesoucí důležité informace pro operátory. Zároveň bylo implementováno několik komponent, jejichž funkcionalita plyne z aktuálních požadavků na repasovací proces. Při návrhu editoru byl kladen důraz na jednoduchost použití, aby byl program intuitivní a snadno použitelný i pro technicky méně zdatné uživatele. Z vytvořených stanic lze poté sestavovat linky pro jednotlivé konstrukční typy a specifikovat výchozí nastavení vstupů stanic. Správa referencí byla přepracována tak, aby zobrazovala vazby mezi referencemi a konstrukčním typem, pro který byly založeny. Byla přidána možnost

nastavit počty a typy komponent vstupujících a vystupujících z procesu z důvodu přepracování způsobu uskladnění komponent. S touto změnou souvisí také úprava editoru reportů tak, aby mohly být zadávány identifikátory vstupních komponent při započetí práce na konkrétní stanici, stejně jako identifikátory výstupních komponent po dokončení stanice.

Dokončením této diplomové práce však není ukončena práce na samotné aplikaci. Jako další kroky v jejím vývoji se jeví implementace:

pokročilých komponent pro editor stanic - Jedná se o specifické komponenty vytvořené na míru pro unikátní použití. Konkrétním příkladem může být parser RTF souborů, které jsou vytvářeny speciálním softwarem použitým při přehrávání softwaru řídících jednotek airbagů. Jedná se o třístránkový dokument, ze kterého je nejdůležitější řádek se sériovým číslem dané ECU a s aktuální verzí softwaru. Aby bylo použití nové komponenty co nejjednodušší a přitom univerzální, měl by být uživatel schopný otevřít vzorový dokument dle svého výběru, a v grafickém editoru myší označit část textu, kterou chce v dokumentu vyhledat (např. text „Serial number“) a poté specifikovat kolik následujících znaků má být do databáze uloženo (označením myší přímo sériového čísla, zadáním počtu znaků apod.). Samozřejmostí je podpora více formátů vstupních souborů.

uživatelsky přívětivého prohlížeče historie komponent - Rozpadem výrobků na jednotlivé komponenty a jejich opětovným skládáním vzniká souvislý neohodnocený graf známý z teorie grafů. Pro zobrazení historie vybrané komponenty se tedy nabízí sestavit a vykreslit uživateli tento graf. Uzly grafu mohou obsahovat číslo zakázky nebo unikátní identifikátor komponenty, které mohou zároveň sloužit jako odkazy pro otevření konkrétní zakázky.

editoru pro snadné vytváření statistik z uložených dat - Grafický editor pro pohodlné sestavení složitějších SQL dotazů pro zobrazení údajů o počtu repasovaných kusů v jednotlivých měsících, procentuálního zastoupení konkrétních hodnot na dané stanici a pro danou referenci a podobně.

Aplikace je v současné době připravena na první testování přímo na repasních linkách. Během vývoje aplikace však došlo ve firmě TRW Frýdlant s.r.o ke změně priorit nových projektů a tak prozatím nebyla na linku nasazena. V době dokončení diplomové práce není datum nasazení stanoveno.

9 Příloha A: Obsah přiloženého CD

Zdrojový kód nové aplikace (projekt pro vývojové prostředí NetBeans IDE 8.1)

ER diagram původní databáze

ER diagram upravené databáze

Diplomová práce ve formátu PDF

Literatura

[1] TVRDÍK, Aleš. *Sledování výrobního procesu repasování*. Liberec, 2013. Technická univerzita v Liberci. Vedoucí práce Tomáš Martinec.

[2] WILKINSON, Leo. Mini: a brief history. *The Telegraph* [online]. **18.11.2013** [cit. 2016-08-03]. Dostupné z: http://www.telegraph.co.uk/motoring/car-manufacturers/mini/10456893/Mini-a-brief-history.html

[3] Remanufacturing: Remanufacturing Overview and History. *Caterpillar* [online]. 2016 [cit. 2016-08-03]. Dostupné z: http://www.caterpillar.com/cs/company/sustainability/remanufacturing.html

[4] TRW. *Požadavky pro vratné díly TRW*. 2016. Dostupné také z: http://img.elit.cz/rgcmedia/s/153/199/43fc6d0df827481cbb3ebe2c2ffedfca.pdf

[5] PROKEŠ, Tomáš. TRW. *Matice kódů konstrukčních typů a zákaznických balení*. 22/6/2015. Frýdlant, 2015. C040CZ-LP02-LF01.

[6] Pareto's principle: the 80-20 rule, pareto's law, or pareto theory. *Businessballs.com* [online]. [cit. 2016-09-03]. Dostupné z: http://www.businessballs.com/pareto-principle-80-20-rule.htm

[7] *Introduction to the Basic Functions of CANape*. Version 1.1. Vector Informatik GmbH, 2012, 14 s. Dostupné také z: http://vector.com/portal/medien/ecu_calibration/canape/CANape_Basics_WhitePaper_EN.pdf

[8] *CANape: Product Information*. V 2.0. Vector Informatik GmbH, 2016, 18 s. Dostupné také z: http://vector.com/portal/medien/cmc/info/CANape_ProductInformation_EN.pdf

[9] *Interface Programming between CANape and MATLAB®*. Version 5.6. Vector Informatik GmbH, 2013/04/11, 46 s. Dostupné také z: http://vector.com/portal/medien/cmc/application_notes/AN-IMC-1-004_Interface_Programming_between_CANape_and_MATLAB.pdf

[10] Using AUTO_INCREMENT. *MySQL 5.7 Reference Manual* [online]. [cit. 2016-08-03]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/example-auto-increment.html

[11] Factory Method Pattern. *OODesign.com: Object Oriented Design* [online]. [cit. 2016-08-03]. Dostupné z: http://www.oodesign.com/factory-method-pattern.html

[12] Decorator Pattern. *OODesign.com: Object Oriented Design* [online]. [cit. 2016-08-03]. Dostupné z: http://www.oodesign.com/decorator-pattern.html

[13] Date on Database: Writings 2000-2006. *Date on database: writings 2000-2006* [online]. New York, NY: Distributed to the book trade worldwide by Springer-Verlag, c2006, s. 107-113 [cit. 2016-08-13]. ISBN 9781590597460.

[14] INFORMATION_SCHEMA Tables. *MySQL Documentation* [online]. 2016 [cit. 2016-08-07]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/information-schema.html

[15] How to Use HTML in Swing Components. *Java Documentation: The Java^{TM} Tutorials* [online]. [cit. 2016-08-13]. Dostupné z: https://docs.oracle.com/javase/tutorial/uiswing/components/html.html

[16] Core Swing. TOPLEY, Kim. *Core Swing: advanced programming* [online]. Upper Saddle River, NJ: Prentice Hall PTR, c2000, s. 426 [cit. 2016-08-14]. ISBN 0130832928. Dostupné z: books.google.cz

[17] VOJTĚCHOVSKÝ, Michal. TRW. *Seznam výrobních operací*. 02. Frýdlant, 2010. C100CZ-LP01-LF06.