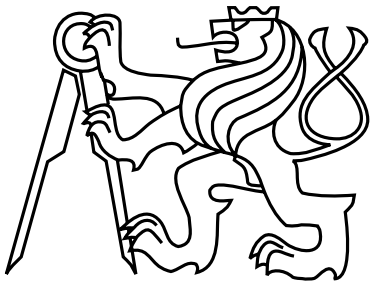
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Grafický editor zabezpečovacího zařízení pro železnici

Jiří Kulovaný

Vedoucí práce: Ing. Pavel Vít

Studijní program: Elektrotechnika a informatika, dobíhající, Bakalářský

Obor: Výpočetní technika

22. května 2011

Poděkování

V úvodu bych chtěl poděkovat svému okolí a blízkým za shovívavost v době psaní této práce. Dále bych chtěl velmi poděkovat Ing. Pavlu Vítovi, vedoucímu práce, který mi zjednodušil práci svými požadavky na časové rozložení práce do delší doby.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Českém Krumlově dne ....

Abstract

The goal of mine bachelor thesis is to create a graphical editor of railway station safety devices. The editor is able to capture topology of railway station safety defices and write it down to XML file. The XML structure of the file is defined by Ing. Pavel Vít in his master's thesis[9].

Application is written in Java language with Swing extension.

Abstrakt

Práce se zabývá vytvořením grafického editoru topologie staničního zabezpečovacího zařízení. Topologii zabezpečovacích komponent zachycenou v diagramu a jejich vlastnosti je schopný zapsat do strukturovaného souboru formátu XML dle definice v diplomové práci Ing. Pavla Víta[9].

Program je napsán v jazyce Java s rozšířením Swing.

Obsah

1 Úvod **1**

2 Popis problému, specifikace cíle **3**

3 Analýza a návrh řešení **5**

3.1 Použité nástroje . . . 5

3.1.1 Technologie . . . 5

3.1.2 Vývojové prostředí . . . 6

3.1.3 Další nástroje . . . 6

3.2 Návrh editoru . . . 7

3.2.1 Vymezení funkčnosti . . . 7

3.2.2 Stavba programu . . . 7

3.2.3 Use Case diagram . . . 9

3.3 Metodika vývoje . . . 11

3.3.1 Zapojení uživatele . . . 11

3.3.2 Průběžné testování . . . 11

3.3.3 Ukončení iterace . . . 12

3.4 Existující nástroje . . . 12

4 Realizace **13**

4.1 Třída EditorGUI . . . 13

4.2 Třída EdWorkspace . . . 14

4.2.1 Komponenty . . . 14

4.2.2 Spoje . . . 15

4.2.3 Zásahy objektů . . . 15

4.2.4 Přibližování a oddalování (Zoom) . . . 16

4.2.5 Generování výstupu . . . 16

4.2.6 Ukládání a nahrávání souboru . . . 16

4.2.7 Historie akcí . . . 17
4.3 Třída EdComponent . . . 17
4.3.1 Třída EdComponentLabel . . . 18
4.3.2 Třída EdComponentVJ . . . 19
4.3.3 Třída EdComponentHQ . . . 19
4.3.4 Třída EdComponentSD . . . 20
4.3.5 Třída EdComponentK . . . 21
4.3.6 Třída EdComponentM . . . 22
4.4 Třída EdConnection . . . 22
4.4.1 Třída EdPoint . . . 23

5 Testování **25**
5.1 Testovaná nádraží . . . 25
5.1.1 Jednoduché nádraží . . . 25
5.1.2 Rozvětvené nádraží . . . 26
5.1.3 Nádraží s odvratovou kolejí . . . 27
5.1.4 Nádraží v Rosicích nad Labem . . . 27

6 Použití editoru **29**
6.1 Návrh topologie . . . 29
6.1.1 Tvorba komponent . . . 29
6.1.2 Práce s komponentami . . . 30
6.1.3 Tvorba spojů . . . 32
6.2 Práce s vytvořenou topologií . . . 34
6.3 Další funkce editoru . . . 34
6.3.1 Skupinový výběr . . . 34
6.3.2 Historie stavů . . . 35

7 Závěr **37**

A Seznam použitých zkratek **41**

B Instalační příručka **43**

C Obsah přiloženého CD **45**

Seznam obrázků

3.1 Use Case diagram 9
3.2 Iterativní cyklus 11

4.1 Komponenta VJ 19
4.2 Komponenta HQ 19
4.3 Komponenta SD 20
4.4 Komponenta K 21
4.5 Komponenta M 22

5.1 Nádraží 1 26
5.2 Nádraží 2 26
5.3 Nádraží 3 27

6.1 Příklad: Úprava komponenty SD 31
6.2 Příklad: spojování 33
6.3 Příklad: skupinový výběr 35

Kapitola 1

Úvod

Diplomová práce [9] Ing. Pavla Víta[1], která rozšiřovala diplomovou práci Ing. Martina Zatrepálka [10], se zabývala vytvořením generátoru řídícího VHDL souboru staničního zabezpečovacího zařízení. Soubor propojuje bloky zařízení dle uživatelského návrhu. Účelem práce bylo napsání programu pro vytvoření nového nádraží, které se nahrává do přípravku FPGA a používá pro další simulace, případně do budoucna v praxi. Program na vstupu přijímá strukturu zabezpečovacích zařízení ve formátu XML. Tento formát byl využit pro jeho přehlednou strukturu bez použití jakéhokoliv nadstandardního editoru.

V současné době neexistuje kvalitní nástroj pro generování vstupu do programu. Aktuálně používaný postup vytvoření XML souboru je pomocí nástroje vytvořeným Ing. Pavlem Vítem v rámci diplomové práce[9]. Nástroj umožňuje vložit hlavičku XML souboru, následné vkládání komponent s ručním nastavením identifikačních čísel vázaných bloků a dokončení vložením patičky uzavírající XML tagy otevřené v hlavičce.

Řešení je však nedostatečné nejen z hlediska časové náročnosti. Uživatel již při vytváření první komponenty musí nastavovat identifikační čísla následujících komponent. To ho nutí mít vypsané vazby mezi komponentami a identifikační čísla někde mimo tento nástroj generující XML. Pokud se přepíše v hodnotě vázaného bloku, je problém zjištěn nejdříve na vstupu programu generujícího VHDL soubor[2]. Pokud však udělá chyb více, je možné, že vytvoří konzistentní soubor a generátor nenahlásí žádnou chybu.

Tyto nedostatky by měla řešit má bakalářská práce. Díky grafickému vykreslení topologie zabezpečovacích zařízení si uživatel může ověřit správnost topologie zabezpečujících prvků. Identifikační čísla jsou generována automaticky, nezávisle na uživateli.

[1] V textu nadále bude odkazována pouze jako diplomová práce s odkazem na literaturu.
[2] V textu nadále bude odkazován jako generátor

Kapitola 2

Popis problému, specifikace cíle

Práce vytváří grafické rozhraní pro sestavení topologie staničních zabezpečovacích prvků a zobrazuje vazby mezi jednotlivými komponentami. Výsledný soubor musí být kompatibilní se strukturou definičního souboru formátu XSD z diplomové práce [9], potažmo s vstupem generátoru.

Cílem mé bakalářské práce je:

- Vytvořit grafické rozhraní, umožňující sestavení topologie zabezpečovacích prvků
- Ošetřit správnost zapojení blokových spojů
- Umožnit generování korektního výstupního souboru ve formátu XML
- Umožnit ukládání aktuálního stavu editoru do souboru
- Připravit několik ukázkových nádraží včetně souboru ve formátu XML vygenerovaného aplikací

Kapitola 3

Analýza a návrh řešení

Tato kapitola se zabývá návrhem editoru. Popisuje použité vývojové prvky a naznačuje rozsah a způsob řešení problémů s vývojem editoru vznikajících. Dále upřesňuje a rozšiřuje realizaci jednotlivých částí editoru a zavádí vývojovou metodiku. Až na první sekci by měla být dostatečně abstraktní, aby se příliš neměnila v případě použití jiných technologií.

3.1 Použité nástroje

Celý projekt provázaný bakalářskými a diplomovými pracemi je charakterizován možností použití na různých operačních systémech. Tuto otevřenost chceme zachovat, což se odráží ve vybraných nástrojích.

3.1.1 Technologie

Java

Práce je implementována v jazyku Java[6]. Jde o objektově orientovaný, interpretovaný jazyk. Interpretovaný je proto, že využívá instrukce metajazyka pro virtuální stroj, do kterých se vytvořený program zkompiluje. Tato množina instrukcí je na cílovém počítači překládána do pseudo-strojového kódu interpretovaného při běhu programu. Díky této metodice podporuje většinu dnes běžně dostupných operačních systémů pro stolní počítače. Java tedy zaručuje platformní nezávislost.

Alternativou při výběru bylo použití objektového jazyka C++. Nebyl vybrán z důvodu návaznosti na diplomovou práci [9], protože se předpokládá následné použití generátoru VHDL souboru, který je v Javě napsán. Jazyk C++ by mělo smysl využít, pokud by rozhodovala rychlost běhu editoru. Obecně jsou interpretované jazyky pomalejší, než jazyky překládané do strojového kódu.

Swing

Pro výběr grafického rozhraní byla opět důležitým faktorem přenositelnost mezi systémy. Měl jsem dobrou zkušenost s rozhraním Swing. Rozhraní nabízí vzhled základního okna grafické aplikace, implementace menu, nástrojové lišty, stavové lišty a pracovní plochy s posuvníky a dalších prvků grafického rozhraní. Základní informace o použití Swingu může čtenář získat z návodů na stránkách s tutoriály[7].

Alternativou k Swingu by mohl být Abstract Window Toolkit (AWT)[1], QT Jambi [2], Standard Widget Toolkit (SWT)[3].

- AWT v podstatě práce využívá, protože je nad ním postavena nadstavba v podobě tříd Swing. Samostatné AWT by mě však nutilo vytvářet již vytvořené komponenty, tudíž jsem ho nevybral.
- QT Jambi nebyl vhodný z důvodu využití programovacích struktur jazyka C++, což by silně znepřehlednilo kód editoru.
- Rozhraní SWT nabízí množinu prvků velmi podobnou prostředí Swingu. Navíc nabízí například implementaci system tray ikony[1]. Tuto funkčnost jsem nepotřeboval, a tak jsem se držel známého vývojového prostředí.

3.1.2 Vývojové prostředí

Po výběru technologií bylo nutno vybrat vývojové prostředí tyto technologie podporující. Bylo vybráno volně distribuované prostředí Netbeans ve verzi 6.9.1. Toto prostředí nabízí výbornou podporu zvoleného grafického rozhraní Swing, dále také podporu nástrojů pro verzování typu SVN, či CVS. Verzování jsem využil kvůli možnosti obnovit kompletní stav zdrojových souborů a tím se vrátit k odzkoušené verzi editoru, pokud sada změn nevedla k potřebnému výsledku. Verzování mi také umožnilo jednoduché externí zálohování editoru mimo mé počítače. Vývojové prostředí dále podporuje automatické generování dokumentace na základě meta-informací obsažených ve zdrojových souborech. Posledním, neméně důležitým důvodem pro výběr prostředí byly mé zkušenosti.

3.1.3 Další nástroje

K verzování z předchozího bodu jsem využil bezplatných služeb webového poskytovatele aplikace typu SVN[8]. K tvorbě Use-Case diagramu a diagramu vývojové metodiky jsem využil aplikaci Enterprise Architect[5].

[1] To jsou ikony, které se v systémech rodiny Windows zobrazují v oblasti hodin.

3.2 Návrh editoru

3.2.1 Vymezení funkčnosti

Editor umožní vytvořit topologii funkčních komponent staničního zabezpečovacího zařízení se závislostmi definovanými v diplomové práci[9]. Tyto komponenty budou obsahovat výstupní body. Každému výstupnímu bodu bude možno přiřadit množiny bran, kdy právě jedna množina bude aktivní a zbylé neaktivní. Aktivní množina bran bude zobrazena na komponentě na pozici výstupního bodu. Bude možné změnit aktivní množinu bran aktivováním jedné z neaktivních množin bran a zároveň deaktivací aktivní množiny bran. Pozice výstupních bodů v zobrazení komponenty budou zaměnitelné, a to jak vertikálně, tak i horizontálně.

Každá brána bude zobrazovat název komponenty povolené k připojení k dané bráně a zároveň stav tohoto spojení. Stavy spojení jsou: připojeno, nepřipojeno. Pokud je spojení již ve stavu připojeno, pak editor nedovolí připojení nového spojení stejného typu k dané bráně. Spojení lze rozdělovat zlomy. Zlomem je myšlen bod ve spoji, který nedefinuje topologii staničního zabezpečovacího zařízení, tj. není koncový bod.

Správnost zapojení se bude automaticky kontrolovat. V případě pokusu o nesprávné zapojení bude uživatel editorem upozorněn.

Pokud komponenta vyžaduje pro generování více informací, než pouze informace o ostatních připojených komponentách, je tato možnost uživateli nabídnuta vhodným způsobem. Například kontextovým menu.

Uživatel bude moci exportovat strukturu použitých zabezpečovacích bloků do formátu XML dle definice diplomové práce[9].

3.2.2 Stavba programu

Práce bude využívat možností objektově orientovaného jazyka. Stavba programu bude tvořena za účelem usnadnění realizace exportu do souboru formátu XML.

Exportní soubor obsahuje seznam komponent, jednoznačně určených unikátními identifikátory komponenty. V každé komponentě je obsahem seznam unikátních identifikátorů komponent vyjadřující jejich vztah s danou komponentou. Editor toto bude reflektovat. Bude uchovávat pole objektů bloků a pole objektů spojovacích čar[2]. Objekty bloků bude možno volně rozmístit po pracovní ploše diagramu. Při spojování bloků se spojovací čáry automaticky uchytí v rámci bloku na definované místo přiřazené dané bráně, toto místo

[2]Dále bude využíván název spoj.

se nazývá úchytným bodem. Úchytné body budou pro potřeby této práce popisovány jako výstupního typu, protože orientace spojení není pro generování rozhodující. Editor však pro svoje vnitřní potřeby orientaci spojů využívá.

Díky objektovému návrhu bude editor jednoduše rozšiřitelný o nové vlastnosti daných objektů, například o nové brány komponent, případně o celé nové komponenty. Pro objekty funkčních bloků bude vytvořena abstraktní třída s definovanými virtuálními metodami nutnými pro funkčnost editor, např. pro generování svého XML výstupu.

Každý funkční blok bude obsahovat informaci svém o vzhledu a bude schopen se vykreslit do grafického uživatelského rozhraní. Také bude obsahovat množinu výstupních bran. Výstupní brána je logická jednotka, dohlížející na povolená spojení v jednu chvíli. Vztah mezi bloky bude znázorněn spojem. Jeho koncové body budou vždy umístěny na úchytném bodu brány zobrazujícím cílovou komponentu.

Stavem editoru je rozmístění komponent a vlastnosti komponent, nastavení výstupních bran, pozice spojů a jejich bodů. Stav editoru bude možno uložit do souboru a následně ze souboru nahrát. Při nahrávání se předchozí stav zruší a bude plně nahrazen stavem novým.

Editor bude umožňovat procházet množinou stavů editoru zpět a vpřed. Množina bude omezená z výkonnostních důvodů. Počet obrazů stavů editoru, kterými bude možno procházet bude 20. K úplnému smazání seznamu stavů dojde v případě vytvoření nového diagramu, nebo v případě nahrání diagramu ze souboru.

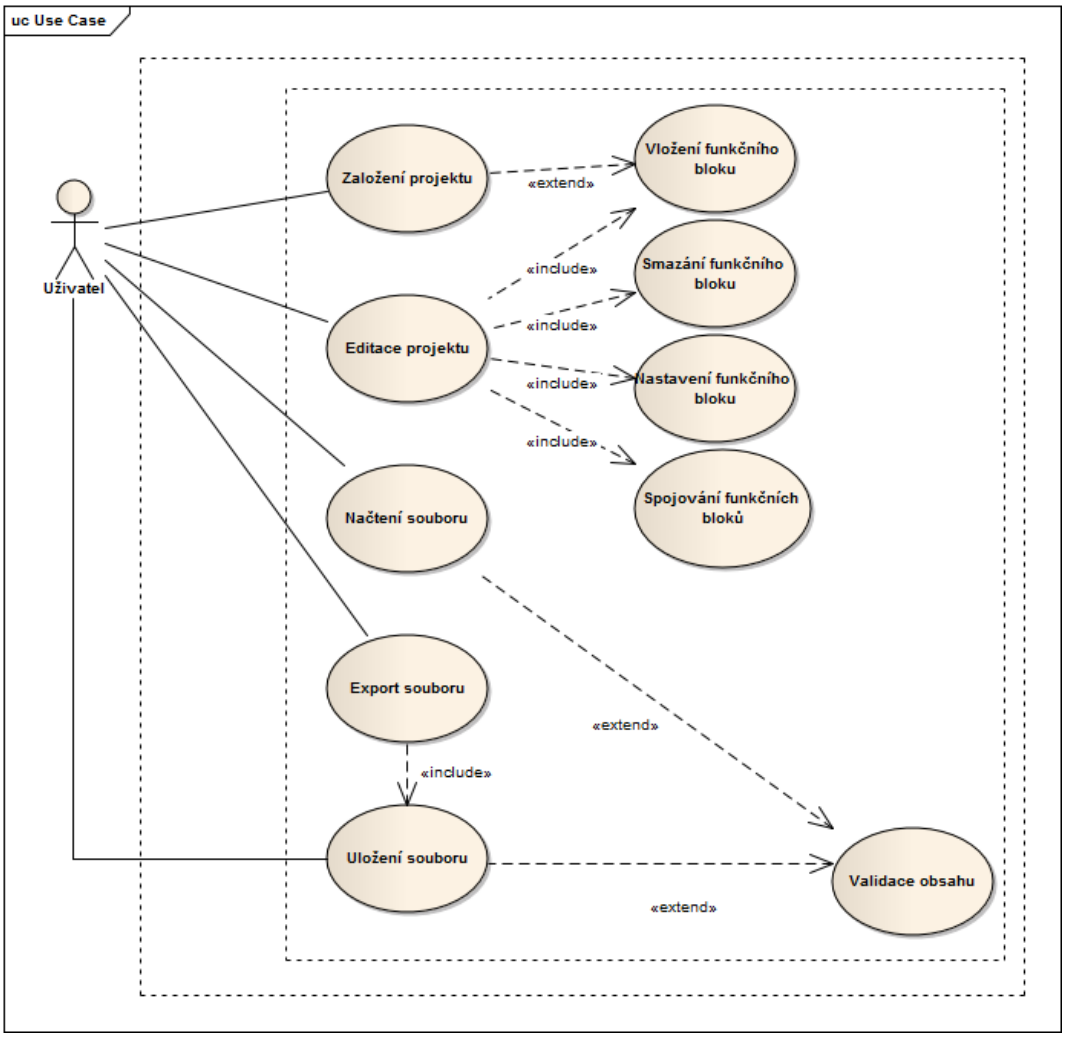
Program umožní generovat a ukládat korektní výstupní soubor ve formátu XML. Tento výstupní soubor bude obsahovat všechny komponenty a jejich vazby. Pokud nebude zapojení výstupních bloků správné, bude uživatel upozorněn a soubor nebude vygenerován.

Lišta programu bude vypisovat jméno otevřeného diagramu. Pokud bude editor ve stavu jiném než přímo po nahrání, uložení nebo vytvoření nového diagramu, pak se tato informace ukáže v liště programu. Zobrazena bude hvězdičkou před jménem otevřeného diagramu.

Bude možné vybrat skupinu komponent a spojů a touto skupinou posunovat po pracovní ploše, pokud bude posunováno jedním z vybraných prvků. Vybrané prvky budou patřičně výrazně označeny. Prvky v této množině bude možné najednou smazat. Pokud je skupinově vybrán pouze jeden ze zlomů spoje, pak se smaže celý spoj.

Součásti grafického rozhraní, u kterých se předpokládá časté použití, budou mít přiřazené události vyvolávané klávesovou zkratkou. Zkratky budou obecně respektovat kombinace běžně zavedené pro grafické aplikace. Menu budou mít přiřazenu kombinaci klávesy *<Control>* s písmenem, zkratky nástrojové lišty budou volané kombinací klávesy *<Alt>* a znaku nebo jiným vhodnějším způsobem.

3.2.3 Use Case diagram



Obrázek 3.1: Use Case diagram

Založení projektu

Editor umožní uživateli vytvořit nový prázdný diagram. Projektu je automaticky přiřazen standardní název *Unnamed.ed*.

Editace projektu

Editor umožní upravovat otevřený diagram modifikací funkčních bloků a jejich spojů:

- vložením nového bloku s možností výběru místa vložení
- smazáním bloku včetně jeho spojovacích čar
- úpravou vlastností existujícího bloku
- přemístěním bloku
- vytvořením spoje bloků, případně i se zlomy
- smazáním spoje bloků
- přidáním a odebráním zlomu do spoje bloků
- vrácením nebo znovu vytvořením stavu editoru
- vybráním více bloků a jejich společným
  - přemístěním
  - smazáním

Uložení souboru

Editor umožní ukládání souboru do vlastního výstupního souboru. Editor ukládá svůj stav ve smyslu 3.2.2. Výsledkem úspěšného dokončení operace je výstupní soubor na disku.

Načtení souboru

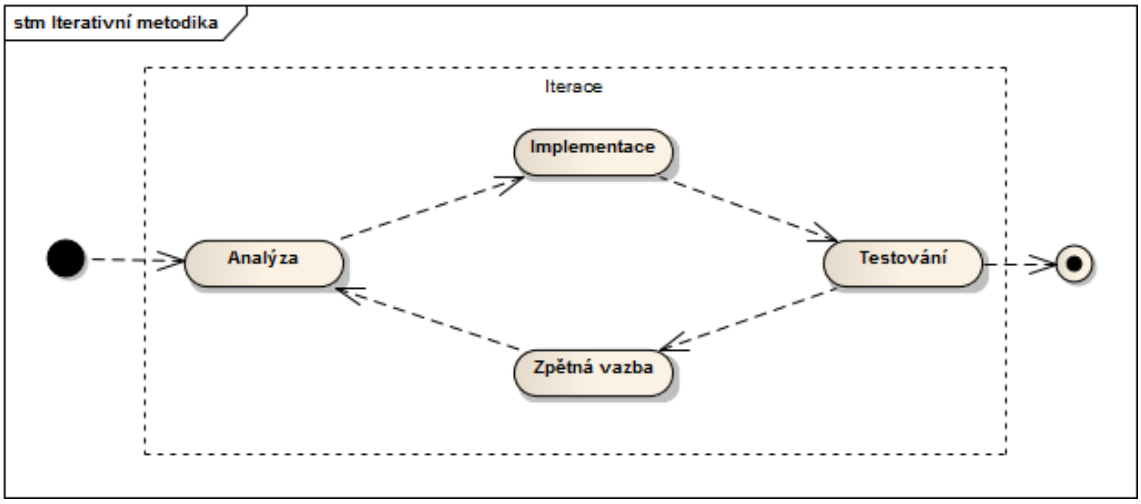
Editor umožní načtení uloženého stavu editoru ze souboru. Před začátkem načítání se resetuje vnitřní stav editoru. Výsledkem úspěšného dokončení operace je pracovní prostředí s komponentami a spoji načtenými z souboru.

Export souboru

Editor umožní uživateli uložit korektní výstupní soubor formátu XML. Pokud chybí vazba mezi bloky nebo není vyplněná některá z jejich požadovaných vlastností, bude uživatel na tento nedostatek upozorněn a export ukončen. Výsledkem úspěšného dokončení operace je vytvořený konzistentní XML soubor.

3.3 Metodika vývoje

Proces vývoje může využívat čistě metodiky agilního vývoje[4] nebo klasické metodiky vývoje. V tomto projektu použiji zlatou střední cestu využitím výhod obou přístupů. Z metodiky agilního vývoje využiji metodu započetí vývoje hned v raných fázích projektu. Z metodiky klasického vývoje pak standardní tvorbu analytických dokumentů. Nebudu tedy považovat zdrojový kód za samodokumentovaný. Důležitou součástí je využití iterativního přístupu k vývoji s co nejmenším vývojovým cyklem viz obr. 3.2



Obrázek 3.2: Iterativní cyklus

3.3.1 Zapojení uživatele

Důležitou částí metodiky vývoje je snaha v maximální možné míře zapojit budoucího uživatele softwaru do všech fází projektu. Iterativní přístup pomůže zefektivnit práci na vývoji tím, že se software nebude muset velkou měrou dodatečně přizpůsobovat změněným požadavkům uživatele.

3.3.2 Průběžné testování

Při vývoji software budou jednotlivé části průběžně podrobeny uživatelskému testování, aby se odhalily chyby dříve, než se na nich postaví další část projektu.

3.3.3 Ukončení iterace

Iterace se opakují, dokud editor neprojde úspěšným koncovým testem. Výstupem každé iterace vývoje editoru je konzultace s uživatelem, shrnutí nově nasbíraných poznatků a promítnutí závěru do další iterace. Iterace by měla být dokončena v kratším časovém úseku, například dva až tři týdny. Schůzky s uživatelem tedy budou navrhnuty dopředu v konstantním cyklu třech týdnů.

3.4 Existující nástroje

V současné době existují dva nástroje pro vytváření struktury vstupního XML souboru aplikace generátoru. Prvním je nástroj z diplomové práce [9] vytvořený v jazyce Java. Druhým je jakýkoliv editor textu. Pod systémy rodiny Windows by to byl například Notepad. Pod systémy rodiny *nix můžeme vybrat například editor Vim.

První, přehlednější editor nabízí grafické rozhraní uzpůsobené tvorbě požadovaného výstupního XML souboru. Lze v něm vložit hlavičku XML souboru stiskem tlačítka, poté sekvenčně vkládat jednotlivé komponenty a ručně nastavovat identifikátory komponent vázaných na brány. Po dokončení vkládání komponent se stiskem tlačítka vloží patička XML a soubor lze uložit na disk. Toto řešení je plně funkční a použitelné. Pouze je mírně nepřehledné a nenabízí možnost dodatečné úpravy jednotlivých vložených komponent a vazeb, natož zpětnou úpravu již vytvořené struktury nádraží. Pokud uživatel používá tento nástroj dle zadání, pak použití tohoto nástroje zaručuje korektnost výstupního souboru formátu XML.

Druhým zmiňovaným nástrojem je textový editor. Ten by od uživatele vyžadoval detailní znalost XML reprezentace jednotlivých komponent a hlavně jejich výstupních bran. Oproti předchozímu způsobu tvorby souboru nabízí dodatečné změny v programu. Ty lze ale pomocí textového editoru aplikovat i po vytvoření souboru prvním způsobem. Není to tedy žádná výhoda tohoto způsobu vytváření struktury XML souboru. Další nevýhodou je možná nekorektnost souboru typu XML. Uživatel by musel ručně kontrolovat soubor vůči jeho XSD specifikaci. To u prvního způsobu není nutné, podmínky jsou podchyceny již v daném editoru.

Kapitola 4

Realizace

4.1 Třída EditorGUI

Základním prvkem rozhraní je třída *EditorGUI*. Rozšiřuje třídu Swingu *JFrame*[1]. Tato třída implementuje standardní grafické rozhraní běžné v novějších operačních systémech. Na systémech rodiny Windows lišta okna editoru obsahuje tlačítka pro minimalizaci, maximalizaci a zavření okna. Dále také kontextové menu pro ovládání pozice a dalších vlastností okna. Do třídy *JFrame* a jejích potomků lze přidávat další prvky grafického rozhraní. Pod lištou okna editoru se objevuje lišta aplikačního menu. V editoru je přidáno

- menu *Soubor* obsahující možnosti:
  - *Nový*
  - *Uložit jako...*
  - *Otevřít*
  - *Exportovat*
  - *Konec*
- menu *Úpravy* nabízející průchod historií stavů editoru.
- menu *Nápověda*:
  - *Nápověda k prostředí* s základními informacemi o užití editoru
  - *O programu...* s základními informacemi o editoru

[1] Všechny třídy z práci zmíněné s prvním písmenem velké J jsou třídy rozhraní Swing. Tato informace dále nebude uváděna.

Dalším prvkem přidaným do rozhraní je *JToolBar*, který realizuje panel nástrojů. Panel nástrojů dává uživateli na výběr prvky, které následně může vkládat do pracovního prostředí, a prvky, u kterých se předpokládá časté využití. Obsahuje pět prvků komponent, které mají červené ohraničení, dále je zde spoj pro vytvoření vazby mezi komponentami. Výběr prvků doplňují ovládací tlačítka pro procházení historií stavů editoru a pro ovládání zoomu pracovní plochy.

Ve spodní části rozhraní je prvek *JStatusBar*, který vypisuje aktuální informace pro uživatele. Při modifikaci zoomu vypisuje aktuální stav v procentech. Při tvorbě a přesunu komponenty nebo bodu spoje zobrazuje pozici objektu a aktuální stav zoomu v procentech.

Posledním a nejdůležitějším prvkem je pracovní prostředí, jemuž se věnuje následující kapitola. To vytváří třída *EdWorkspace* popsaná v další sekci. Pracovní prostředí se dynamicky přizpůsobuje velikosti okna editoru.

4.2 Třída EdWorkspace

Tato třída představuje pracovní prostředí editoru. Skládá se ze dvou důležitých prvků. Za prvé třída rozšiřuje třídu *JScrollPane*. Objekty třídy *JScrollPane* jsou spíše průzorem na vykreslovanou plochu. Vykreslovaná plocha tedy může být libovolně velká a pokud přesáhne hranice, které je *JScrollPane* schopno vykreslit, tak se objeví posuvníky. Ty umožní uživateli posun viditelného výseku pracovní plochy. Pracovní plocha je realizována objektem třídy *JPanel*, který standardně umožňuje vkládání dalších komponent z rodiny Swing a jejich vykreslování. Pro potřeby editoru nebylo této funkčnosti využito. Metoda paint, starající se o vykreslování objektů do plochy vložených, byla přetížena a vykreslování řídím sám. Díky rozdělení plochy a průzoru bylo možné rozšiřovat pracovní plochu dynamicky s polohou komponent a zlomů. Pokud uživatel umístí nebo přemístí komponentu, případně zlomový spoj, editor kontroluje nejvzdálenější vykreslovaný objekt a dle jeho vnějších hranic přizpůsobuje velikost objektu *JPanel*. Horizontální a vertikální posuvníky se tedy zobrazují v závislosti na pozici nejvzdálenějších komponent od levého horního rohu pracovní plochy. Pokud je velikost *JPanel*u menší, než velikost průzoru, posuvníky v *JScrollPane* zmizí.

4.2.1 Komponenty

Základní funkcí *EdWorkspace* je práce s komponentami a jejich spoji. Komponenty dle definice diplomové práce [9] jsou VJ[2], HQ, SD, K, M. Bližší informace ke komponentám jsou

[2] Jméno komponenty bylo na doporučení vedoucího práce zkráceno z původního VJZD.

v sekci 4.3.2 a následujících. Po vybrání komponenty z nástrojové lišty lze vytvářet komponenty daného typu. Tak se děje, dokud uživatel nevybere jiný nástroj z nástrojové lišty, nebo pokud neukončí tvorbu klávesou *<Escape>*, případně pravým tlačítkem myši. Po dokončení vkládání komponent do pracovního prostředí má uživatel možnost pravým tlačítkem v zásahové ploše, popsané v sekci 4.2.3, vyvolat kontextové menu komponenty. Zde lze nastavit její vlastnosti, případně ji smazat.

4.2.2 Spoje

Vložené komponenty lze propojovat spojem. Krajní body spoje se po kontrole správnosti zapojení uchytí na místech, která jim komponenta určí. Mezi krajními body je možno vytvořit libovolný počet zlomů. Vytváření zlomů reaguje na klik levým tlačítkem myši při stisknuté klávese *<Control>* v zásahové oblasti spoje. Rušení zlomů se provede v zásahové oblasti zlomu, popsané v sekci 4.2.3, klikem levým tlačítkem myši při stisknuté klávese *<Control>*. Mazání je přiřazena vyšší priorita než tvorbě zlomů. Pokud tedy uživatel v blízkosti zlomu stiskne tuto kombinaci dvakrát za sebou, nejdříve se zlom smaže a poté se vytvoří nový. Přes kontextové menu lze spoj, stejně jako komponentu, mazat. Zásah spoje pro otevření kontextového menu se vyhodnocuje trefou zásahové plochy popsané v další sekci.

4.2.3 Zásahy objektů

U objektů pracovního prostředí se vyhodnocuje zásah v případě, že uživatel stiskne levé tlačítko myši s kurzorem nad zásahovou plochou objektu. Pro každý typ objektu je zásahová plocha určena zvlášť. Komponenty mají zásahovou plochu shodnou s plochou grafického znázornění komponenty v pracovní ploše.

Spoje mají dva druhy zásahové plochy.

- První zásahovou plochou je zásah bodu, ze kterého se spoj skládá. Koncový bod nebo zlom je zasažen, pokud uživatel stiskne levé tlačítko myši s kurzorem ve čtverci s šířkou definovanou tolerancí a středem na pozici bodu.
- Druhá zásahová plocha se nachází v oválné oblasti okolo čar spojů.

Poznámka: Díky tomuto přístupu je plocha užší u zlomových a koncových bodů a nejširší ve středu úsečky. Řešení je vhodné kvůli velké hustotě spojnic v okolí výstupních bodů komponent. Zde chceme pro každou komponentu menší plochu pro zásah, než ve středu úsečky, kde se nepředpokládá další blízká čára.

4.2.4 Přibližování a oddalování (Zoom)

Pracovní plocha podporuje metodu přibližování a oddalování. Při použití této funkce se zachovává poměrná velikost komponent stejně jako jejich pozice vůči počátečnímu bodu. Funkce by měla usnadnit přehled nad velkým diagramem a případně umožnit práci s programem i uživatelům s lehce sníženou zrakovou schopností. Při pohybu komponent v různých úrovních přiblížení by se komponenty nesprávně umisťovaly. Z tohoto důvodu bylo použito relativní umístění, kdy se přemístění grafického objektu při určitém přiblížení přepočítává do jednotného standardního pozicového systému a při vykreslení se vypočítává aktuální poloha a velikost v závislosti na zoomu. Zoom ovlivňuje nejen celkovou velikost komponent, ale i tloušťku čar, velikost zlomů a velikost písma na jednotlivých komponentách.

4.2.5 Generování výstupu

Dalším úkolem pracovní plochy je generování výstupu ve formátu XML. Funkce využívá pozici komponenty v poli jako její unikátní identifikační číslo. Pro účely generování vytváří *StringBuffer*, realizující kontext, do kterého vloží hlavičku výstupního souboru typu XML. Následně kontext předá všem komponentám pracovní plochy, aby do něj zapsaly své informace. Každá komponenta tedy vkládá pouze informace pro ní důležité, včetně unikátních identifikátorů odkazovaných komponent. Nakonec funkce do kontextu vloží patičku a nabídne uživateli uložení. Při generování se provádí kontrola správnosti a úplnosti zapojení. V případě problému je na tuto skutečnost uživatel upozorněn informačním oknem se jménem a unikátním identifikátorem komponenty. Pro tuto chvíli jsou komponentách zobrazeny unikátní identifikátory.

4.2.6 Ukládání a nahrávání souboru

Třída realizuje funkce ukládání stavu editoru do souboru. K tomu je využita možnost serializování objektů do souboru. Každá serializovaná třída musí implementovat interface `java.io.Serializable`. Pro potomky třídy EdComponent vypadá hlavička takto:

```
public abstract class EdComponent implements Serializable
```

Takováto jednoduchá hlavička nám umožní zapsat obsah třídy do souboru. Nutnost implementace vyplývá z toho, že programátor musí vytvářet danou třídu při dodržení pravidel pro serializaci. Pokud některé objektové proměnné nechceme serializovat do souboru, pak jim přidáme předponu *transient*. To jsem udělal například u proměnné *workspace*. Tím jsem omezil serializaci pouze na objekt komponenty a jeho vazby, což je vše, co potřebuji.

Pro jednoduché uložení stavu editoru jsem využil toho, že se všechny komponenty a spoje nachází v dynamickém poli *ArrayList*. Stačilo serializovat dynamická pole a tím byla zajištěna serializace všech objektů včetně funkčních vazeb. Předpokládal jsem, že by proti využití této funkčnosti mohla mluvit velikost výstupního souboru. Ten se však pro nádraží na obr. 5.3 pohybuje v okolí 32kB, což v dnešní době není mnoho.

Následná implementace nahrání stavu editoru ze souboru byla jednoduchou záležitostí. Nejprve editor zruší aktuální stav. Poté vyvolá dialog výběru souboru. Po vybrání souboru je dynamické pole obsahující komponenty nahrazeno dynamickým polem obsahujícím komponenty ze souboru. Dynamické pole obsahující spoje je také nahrazeno dynamickým polem spojů ze souboru. Poté je provedena nutná inicializace neserializovaných součástí tříd (například listenerů pro vyvolání kontextového menu). Dále také ošetření některých vnitřních proměnných jako je identifikátor vybrání komponenty.

Poznámka: Pro dialog výběru souboru jsem vytvořil třídu *EdFileFilter* rozšiřující *FileFilter*, která umožňuje dialogu zúžit výběr souborů na koncovku *.ed*. Uživatel si však může vybrat i filtr umožňující výběr ze všech souborů.

4.2.7 Historie akcí

EdWorkspace implementuje historii akcí. Při každé úspěšně dokončené akci se volá metoda *saveState*, která serializuje aktuální stav, dle definice v sekci 3.2.2, do dynamického pole stavů. Položky seznamu stavů tentokrát nejsou souborového typu, jako to je u metody uložení. Jsou to objekty typu *ByteStream*. Procházení těmito stavy je pak analogické nahrávání stavu ze souboru.

4.3 Třída EdComponent

Třída dodává rozhraní pro jednotlivé děděné třídy a zároveň realizuje základní funkce shodné pro všechny komponenty. Například pozicování a získání základních informací o objektu.

Nejdůležitějšími vlastnostmi komponenty je její schopnost vykreslit se na požadované pozici, přijmout spojení a zapsat svoji strukturu do datového toku. Při umisťování editor vypočítá pozici dle definice v podkapitole 4.2. Při vykreslení komponenty tato třída implementuje pouze hrubý vzhled. Vykreslí objekt o velikosti definované při tvorbě objektu a do středu vykreslí název komponenty. Funkčnost přidání komponent, stejně jako zápis struktury do datového toku, si realizuje každá děděná třída sama.

Každá komponenta tedy musí být schopna generovat svůj výstup do kontextu. Hloubku zanoření v XML souboru není potřeba řešit díky sériovému zapisování komponent za sebe. Komponenta pouze vloží informace, které ji definují, a informace o připojených komponentách. Zbytek informací připojuje třída *EdWorkspace*

Komponenty mají své kontextové menu. To se vyvolá pravým tlačítkem myši v zásahové zóně komponenty. Zásahová zóna je popsána v sekci 4.2.3. Kontextové menu si řídí třída děděné komponenty sama. Může obsahovat činnosti:

- nastavení výstupních bodů
- horizontálního a vertikálního otočení komponenty
- nastavení dalších vlastností komponenty
- nastavení komentáře
- smazání komponenty

Pokud vybereme volbu horizontálního nebo vertikálního otočení komponenty, pak se zamění výstupní body dle popisu v sekci 4.3.1.

Barva ohraničení komponenty určuje její vnitřní stav. Můžou nastat tři možnosti:

- Pokud je komponenta vybraná skupinovým výběrem, vybarví se modře. Tato barva má prioritu před následujícími možnostmi.
- Komponenta může mít správně obsazené výstupní brány, pak je ohraničení vykresleno zelenou barvou.
- Poslední možnost zbývá pro neobsazené některé nutné výstupní brány. V takovém případě je barva ohraničení komponenty červená.

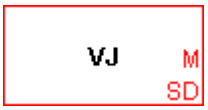
4.3.1 Třída EdComponentLabel

Třída *EdComponent* v sobě ještě ukrývá důležitou součást, třídu *EdComponentLabel*. Ta realizuje vykreslování bran včetně jejich obsazenosti. Každý z rohů komponenty představuje umístění pro výstupní bod. Každý výstupní bod obsahuje dvě pozice pro možný spoj.

Jako příklad uvedu pravý dolní roh komponenty. Výstupní bod obsahuje horní a dolní pozici viz obr. 4.1. Díky tomuto rozdělení bylo snadné implementovat otáčení komponent vertikálně a horizontálně, kdy se pouze zamění pozice příslušných výstupních bodů. Tím se výstupní body vykreslují do správného rohu komponenty.

Brána se zabarvuje dle jejího stavu připojení, podobně jako komponenta. Zelená je pro připojenou bránu, červená pak pro nepřipojenou. Na rozdíl od okraje komponenty zde modrá barva značící výběr není implementována.

4.3.2 Třída EdComponentVJ



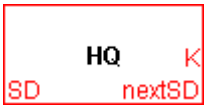
Obrázek 4.1: Komponenta VJ

- Citace z [9]:„Blok je určen k vjezdu vlaků do nádraží a je napojen na vjezdové návěstidlo a jeho předzvěst, kde zobrazuje povolení k vjezdu a maximální rychlost jízdy.“
- Citace z [9]:„Blok umožňuje stavění cesty. Lze z něho nebo do něho postavit vlakovou cestu. Cestu vytvořenou z tohoto bloku lze pouze tímto blokem zrušit.“

Na obr. 4.1 je pohled na komponentu VJ v editoru. Je na ní vidět automaticky nastavený výstupní bod. Množina bran výstupního bodu je definována typy bran M a SD. Dále je možné přidat bránu typu HQ. Ta je umístěna do horního pravého rohu komponenty.

Pro generování musí být brány typu M a SD připojeny spojem k patřičné komponentě. Pokud je přidána i brána typu HQ, pak i ta musí být připojena.

4.3.3 Třída EdComponentHQ



Obrázek 4.2: Komponenta HQ

- Citace z [9]:„Blok ovládá hlavní odjezdové návěstidlo.“
- Citace z [9]:„Tento blok je určen ke stavění cesty. Také lze z něj nebo do něj postavit a zrušit vlakovou cestu stejně jako u bloku VJZD.“

Na obr. 4.2 je pohled na komponentu HQ v editoru. Jsou na ní vidět automaticky nastavené výstupní body.

- Množina bran prvního výstupní bodu je definována množinou obsahující pouze jednu bránu typu SD
- Množina bran druhého výstupního bodu je definována množinou s bránami typu K a nextSD. Množinu bran druhého výstupního bodu lze zaměnit za množinu bran typu VJ a M.

Pokud přímo za komponentou HQ nenásleduje komponenta SD, je možné tuto bránu typu SD nechat nepřipojenou. Brána typu nextSD také může zůstat nepřipojena, pokud první blok HQ následující blok K připojený k bráně typu K není následován blokem SD.

Tato komponenta je jedinou výjimkou kde nemusí být všechny zobrazené brány připojeny k nějaké další komponentě. Tato výjimka se týká pouze bran SD a nextSD.

4.3.4 Třída EdComponentSD

Obrázek 4.3: Komponenta SD

- Citace z [9]:„Blok znázorňuje výhybku, kontroluje její nastavení směru jízdy a dorazů kolejiště. Generuje příkazy pro přestavění výhybky.“
- Citace z [9]:„Dohlíží nad odvratem výhybky, pokud za ní následuje jiná výhybka.“
- Citace z [9]:„Kontroluje směr jízdy.“

Na obr. 4.3 je pohled na komponentu SD v editoru. Jsou na ní vidět automaticky nastavené výstupní body.

- Množina bran prvního výstupního bodu je definována typy VJ a M. Ty lze zaměnit za
  - množinu obsahující brány typů VJ a HQ

  - množinu obsahující brány typů HQ a nextHQ
  - množinu obsahující pouze jednu bránu typu SD.

- Druhý výstupní bod má standardně nastavenou množinu s bránami typů HQ a nextHQ. Další možnou množinou tohoto výstupního bodu je množina obsahující jednu bránu typu SD.

- Třetí výstupní bod má množiny bran shodné s druhým výstupním bodem, tedy má standardně nastavenou množinu s bránami typů HQ a nextHQ. Další možnou množinou tohoto výstupního bodu je množina obsahující jednu bránu typu SD.

Dále je na obrázku vidět rozdílná velikost komponenty SD od ostatních komponent. Dvojnásobná výška je dána počtem výstupních bodů. u této komponenty to jsou již popsané tři výstupní body, kdežto u ostatních to jsou body pouze dva.

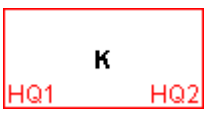
Pokud je některý z výstupních bodů nastavený na množinu bran typu HQ a nextHQ, pak HQ značí blok odjezdového návěstidla přímo za blokem výhybky. Blok nextHQ znázorňuje další blok HQ v sekvenci zařízení.

Kolej, která leží mezi bloky připojenými od prvního na druhý výstupní bod, představuje směr průjezdu rovně. Kolej umístěná mezi prvním a třetím výstupním bodem představuje odbočující směr výhybky.

V kontextovém menu komponenty lze najít speciální položku *Nastavení*. V tomto nastavení máme možnost nastavit průjezdové rychlosti pro směr rovně a pro odbočující směr. Standardně je odbočujícím směrem směr vlevo. Pokud chceme změnit směr vpravo, použijeme vertikální záměnu bran popsanou v sekci 4.3. Změna pozic výstupních bodů je pouze optická. V generovaném výstupu se nic nemění.

Pro generování musí být všechny aktuálně vybrané množiny bran obsazeny.

4.3.5 Třída EdComponentK



Obrázek 4.4: Komponenta K

- Citace z [9]:„Blok kontroluje obsazenost úseku koleje.“

- Citace z [9]:„Dohlíží na směr jízdy vlaku a vlak zde může ukončit jízdu.“
- Citace z [9]:„Vlak zde může zastavit a změnit směr jízdy.“

Na obr. 4.4 je pohled na komponentu K v editoru. Jsou na ní vidět automaticky nastavené výstupní body. První výstupní bod, stejně jako druhý, obsahuje pouze bránu typu HQ. Jejich pozice jsou zaměnitelné pro zvýšení pohodlnosti práce uživatele editoru.

Pro generování musí být obě brány obsazené.

4.3.6 Třída EdComponentM

Obrázek 4.5: Komponenta M

- Citace z [9]:„Blok kontroluje obsazenost úseku koleje, který mu přísluší.“
- Citace z [9]:„Dohlíží na správnost směru jízdy vlaku.“

Na obr. 4.5 je pohled na komponentu K v editoru. Jsou na ní vidět automaticky nastavené výstupní body. První výstupní bod je pevně nastaven na typ VJ. Druhý výstupní bod má nastaven bránu na typ SD. Je možné ho zaměnit za bránu typu HQ.

Pro generování musí být všechny vybrané brány obsazené.

4.4 Třída EdConnection

Poslední, ale neméně důležitou třídou ve výběru je třída realizující spoje komponent. Spoj se skládá z koncových bodu, zlomových bodů a spojujících čar. Počáteční bod spoje může být založen pouze na komponentě a spoj musí být opět na nějaké komponentě ukončen. Po dokončení tvorby spoje se kontroluje správnost propojení. Pokud komponenty dané spojení nepodporují, je spoj smazán a uživatel upozorněn. Čáry spojů jsou vykreslovány vždy mezi dvěma body. Pro obsluhu bodů je zde vnořena třída *EdPoint*.

Spoj lze dodatečně upravovat přidáním a odebráním bodu zlomu a pohybem s body, které jí tvoří. Vytvoření bodu zlomu na vytvořeném spoji řeší funkce *splitLine*. Nejprve zkontroluje zásah spojnice, dle popisu v sekci 4.2.3, pokud je spoj zasažen, pak je do něj vložen nový bod.

4.4.1 Třída EdPoint

Každý bod spoje má dvě důležité vlastnosti:

- reálnou pozici
- pozici pro grafické vykreslení

  Reálnou i grafickou pozici nastavuje uživatel kliknutím, či přemístěním bodu. Pozici pro grafické vykreslení si editor přizpůsobí v případě koncového bodu přichyceného k bráně. Pokud uživatel pracuje s bodem, vyhodnocuje se jeho pozice pro grafické vykreslení. Reálná pozice je používána pouze pro kontrolu zásahu komponenty.

  Bod, stejně jako komponenta, pracuje s pozicovým systémem nezávislým na aktuálním zoomu.

Kapitola 5

Testování

Hlavní testování editoru bylo provedeno primárně postupem definovaným v diplomové práci[9]. Byly vytvořeny čtyři diagramy nádraží s propojením zvolených komponent. Z nich vyexportován soubor typu XML, ten byl kontrolován validátorem a následně byl kontrolován vůči kódu z diplomové práce[9]. V uživatelském rozhraní bylo testováno ošetření možných výjimečných stavů. Testování uživatelského rozhraní probíhalo v souladu s metodikou vývoje viz. podkapitola 3.3. Testování bylo opakované v jednotlivých iteracích vývoje editoru.

Při každé iteraci se při testování objevily menší chyby v logice programu. Příkladem budiž nenavázaný objekt třídy EdComponentLabel pro bránu typu SD u prvního výstupního bodu. V 5.1.3 se tedy nevybarvilo spojení jako připojené.

Velkou výhodou opakovaného uživatelského testování bylo dodatečné specifikování funkčnosti, které by mělo zvýšit uživatelskou přívětivost. Největší úpravou během vývoje bylo přidání skupinového výběru a s tím spojených procesů skupinového posuvu a skupinového mazání.

Ve fázi testování docházelo ke zpětné vazbě nejen u chyb editoru. Bylo také přizpůsobováno ovládání grafického rozhraní. Příkladem může být přidání zlomu do spoje. Původní implementace počítala pouze s přidáním zlomu do spoje pomocí kombinace klávesy *<Control>* s stiskem levého tlačítka myši. Po uživatelském testování byla tato kombinace doplněna ještě o postup dvojitého stisku levého tlačítka myši. Každý z postupů se byl navržen na základě uživatelské zkušenosti různých aplikací.

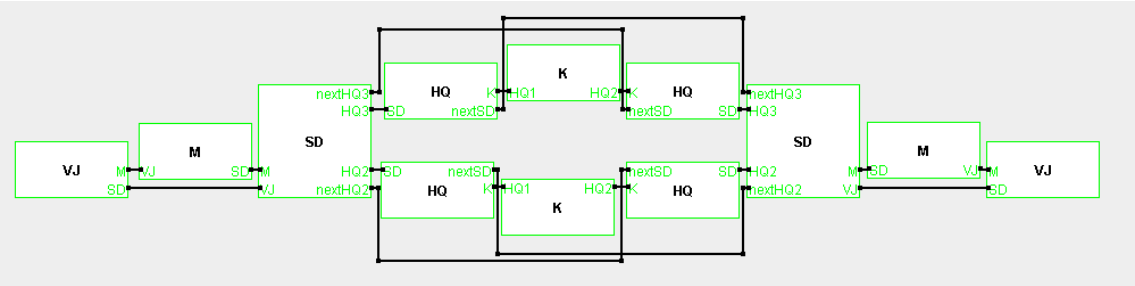
5.1 Testovaná nádraží

5.1.1 Jednoduché nádraží

Citace z [9]:„Prvním vytvořeným nádražím bylo základní nádraží se dvěma kolejemi.“ [9]

Nádraží, zobrazené na obr 5.1, obsahuje základní prvky jednoduše spojené za sebou. Jde o nádraží s dvěma kolejemi bez hlavního odjezdového návěstidla u vjezdu.

Soubory nádraží je možno nalézt na disku CD pod identifikací `nadrazi1`.



Obrázek 5.1: Nádraží 1

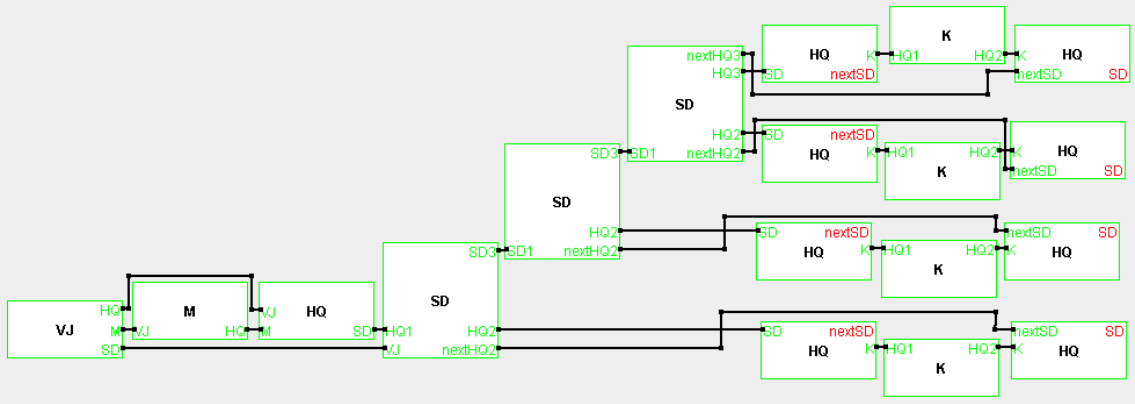
5.1.2 Rozvětvené nádraží

Citace z [9]:„Bylo vytvořeno složitější nádraží, které obsahovalo čtyři staniční koleje a jednu vjezdovou.“ [9]

Nádraží, zobrazené na obr 5.2, bylo zařazeno hlavně z důvodu ukázky případu, kdy nemusí být všechny brány obsazeny. Tato výjimka se týká všech komponent typu HQ, kromě první za vjezdem.

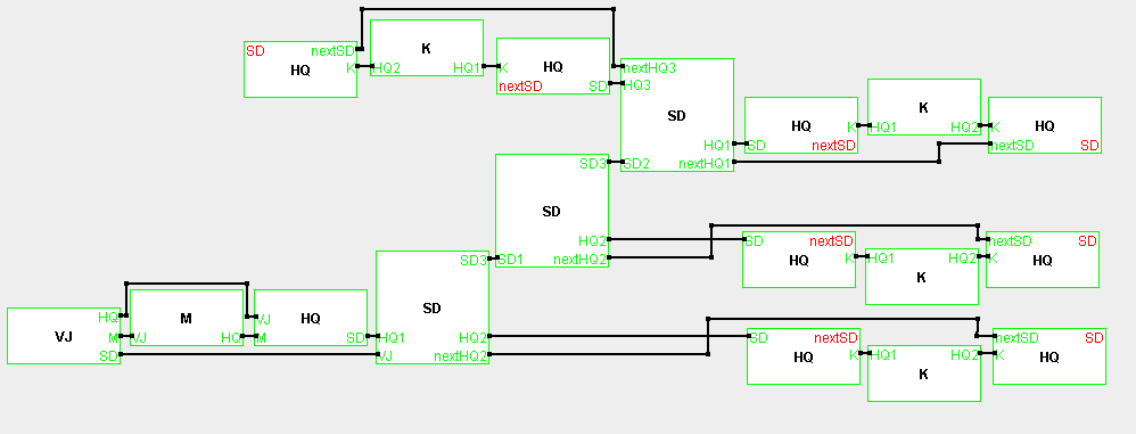
Soubory nádraží je možno nalézt na disku CD pod identifikací `nadrazi2`.

Na obrázku 5.2 jsou komponenty co nejblíže u sebe pro potřeby tohoto dokumentu. Uživatel má neomezenou pracovní plochu, tedy může rozestavení komponent zlepšit.



Obrázek 5.2: Nádraží 2

5.1.3 Nádraží s odvratovou kolejí



Obrázek 5.3: Nádraží 3

Toto nádraží, zobrazené na obr 5.3, bylo v diplomové práci [9] primárně z důvodu testování složitějšího generování VHDL souboru. Pro úplnost jsem ho zařadil i do své práce. Generování výstupu editoru se v ničem neliší při různé složitosti generování kódu v generátoru VHDL. Důležité jsou pouze vazby, ne pozice a otočení komponent.

Soubory nádraží je možno nalézt na disku CD pod identifikací `nadrazi3`.

5.1.4 Nádraží v Rosicích nad Labem

Poslední je nádraží z diplomové práce[9], u kterého se Ing. Pavel Vít inspiroval reálným rozložením staničního zařízení v zastávce Rosice nad Labem. Soubory nádraží lze nalézt na disku CD, obrázek zde není kvůli místu přiložen. Na nádraží se nachází pět kolejí. Nádraží je podobné nádraží z obr 5.2 s tím rozdílem, že za první vyhybkou na druhém výstupním bodu pokračuje další výhybka, která rozšiřuje počet kolejí ze čtyř na pět. Pokud bychom procházeli nádraží zleva doprava, pak vidíme, že se všechny koleje rozpojují přesně stejným způsobem jako se spojují. Jde tedy o palindromické nádraží.

Soubory nádraží je možno nalézt na disku CD pod identifikací `nadrazi4`.

Kapitola 6

Použití editoru

Po spuštění editoru se uživateli zobrazí hlavní obrazovka editoru s vytvořeným prázdným diagramem nádraží. Nyní je možné vidět hlavní prvky:

- menu
- nástrojovou lištu s komponentami
- pracovní plochu s diagramem
- stavovou lištu

Po spuštění je tedy možné rovnou začít navrhovat topologii staničního zabezpečovacího zařízení. Tento stav lze kdykoliv později obnovit výběrem menu *Soubor* položka *Nový*.

6.1 Návrh topologie

6.1.1 Tvorba komponent

Pokud máme čistou pracovní plochu, můžeme začít s návrhem diagramu staničního zabezpečovacího zařízení. Nacházíme se v základním stavu editoru. Nyní je důležité seznámit se s nástrojovou lištou. Na ní se nachází komponenty, které můžeme vkládat na pracovní plochu, a také spojová čára, kterou můžeme spojovat již vytvořené komponenty. Popis jednotlivých komponent je uveden v kapitole 4.

Výběr komponenty se provede stiskem příslušného tlačítka na nástrojové liště. Tím se editor přepne do módu vytváření komponent. V tomto módu je každý stisk levým tlačítkem myši na pracovní plochu vyhodnocován jako pokyn k tvorbě komponenty na uvedeném místě. Mód lze opustit stiskem klávesy *<escape>* nebo stiskem pravého tlačítka myši. Střed

komponenty je standardně centrován na aktuální pozici myši. Jedinou výjimkou jsou hraniční části plochy, které mají jednu z souřadnic pozice menší než nula. Pokud je tato podmínka splněna, pak je nahrazen právě nulou. Souřadnice jsou zapisovány do stavové lišty editoru po vytvoření komponenty a během přesunu komponenty z místa na místo.

Jak bylo naznačeno v předchozím odstavci, lze komponentu přesouvat na jinou pozici. Přesunu docílíme stlačením levého tlačítka myši s kurzorem nad zásahovou plochou komponenty, popsanou v sekci 4.2.3, a následným tažením kurzoru po pracovní ploše. Akci dokončíme uvolněním tlačítka myši na souřadnicích, na kterých chceme komponentu ukotvit. Při tažení nelze přemístit komponentu na pozici s zápornými souřadnicemi.

Příklad: Vybereme komponentu typu HQ v nástrojové liště. Všimněme si, že při přesunutí kurzoru nad pracovní plochu se změní na čtverec s nápisem HQ naznačující vytvářenou komponentu. Zacílíme střed nové komponenty někam ke koordinátům [5,5][1] a stiskneme levé tlačítko myši. Komponenta se celá umístí na pracovní plochu. Nyní vytvoříme druhou komponentu co nejvíce v pravém dolním rohu kliknutím dovnitř pracovní plochy tak, že je kurzor co nejvíce v pravém dolním rohu. Nově vytvořená komponenta bude částečně za rohem průzoru a zároveň ještě bude překryta právě zviditelněnými posuvníky, protože se velikost pracovní plochy dynamicky zvětší. Nyní tedy naše pracovní plocha obsahuje dvě komponenty typu HQ. Obdobným způsobem můžeme vytvořit komponenty dalších typů. Pro potřeby příručky vložíme někam na pracovní plochu ještě jednu komponentu typu SD.

Poznámka k příkladu: V příkladu jsme si ukázali zarovnání komponenty na koordináty [0,0] při tvorbě první komponenty mimo pracovní plochu. Při tvorbě druhé komponenty jsme si ukázali automatické přizpůsobení pracovní plochy velikosti diagramu. Dále také udržení módu vytváření komponent mezi přidáváním jednotlivých komponent typu HQ.

6.1.2 Práce s komponentami

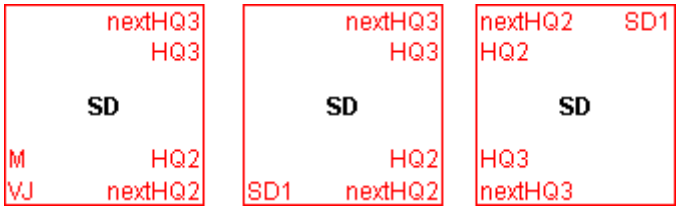
Poté, co máme vytvořené první komponenty si popíšeme jejich vlastnosti a možnosti práce s nimi. Základní vlastností komponenty je stav jejích bran. Aktivní množiny bran jsou vykresleny na komponentě, s popiskou, kterou cílovou komponentu je možné připojit k dané bráně.

Pokud je brána vybarvena červenou barvou, znamená to, že je neobsazena a lze danou komponentu připojit. Brána vybarvená zelenou barvou znamená připojenou komponentu

[1] Pozice [0,0] je v nejlevějším a nejhornějším rohu pracovní plochy. Pozor - není vždy shodný s levým horním rohem průzoru *JScrollPanel*.

a zákaz připojení další komponenty k bráně. Brány vykreslené blízko sebe reprezentují jednotlivé výstupní body. Výstupním bodům lze nastavovat jejich množiny bran, postup je popsán v následujícím odstavci.

Pravým tlačítkem na komponentě se vyvolá její kontextové menu. V něm se nachází prostředky pro úpravy komponent. Mezi prvními je nastavení skupin bran výstupních bodů. Každý výstupní bod má jednu rozbalovací nabídku, ze které lze nastavit aktuální výběr bran. Další možností kontextového menu je otočení komponenty vertikálně nebo horizontálně. Slouží k vzájemnou záměnu skupin výstupních bran po dané ose. Pro komponentu typu SD je jako následující implementována položka *Nastavení rychlosti*. Zde se nastavují rychlosti průjezdu směrem rovně a směrem odbočky. Poslední položkou je možnost smazání komponenty. Při mazání komponenty jsou smazány i všechny její spoje.



Obrázek 6.1: Příklad: Úprava komponenty SD

Příklad: Jako příklad může být úprava prvního výstupního bodu komponenty typu SD. Úvodní stav komponenty SD je zobrazen jako první na obr. 6.1. Stiskem pravého tlačítka myši nad komponentou typu SD vyvoláme kontextové menu. Zde vybereme *První výstupní bod* a dále *SD*. Tím jsme úspěšně změnili výstupní bod na bránu typu SD. Kontrolou nám budiž pohled na komponentu, na které je v levém dolním rohu červeně vykreslený nápis *SD* stejně jako na druhé komponentě na obr. 6.1. Nyní můžeme tento výstupní bod otočit do pravého horního rohu. To uděláme postupným převrácením výstupních bodů horizontálně a vertikálně. Otevřeme kontextové menu a vybereme *Otoč horizontálně*. Znovu otevřeme kontextové menu a vybereme *Otoč vertikálně*. Výsledkem operace je tedy komponenta převrácená oběma směry, jak je vidět na třetí komponentě obr. 6.1. Dále u této komponenty nastavíme její vlastnosti průjezdové rychlosti a komentář. Obojí nastavíme opět přes kontextové menu komponenty.

- Nejdříve vybereme *Nastavení rychlosti*. Otevře se nám dialogové okno obsahující dvě textová pole a potvrzující tlačítko. *Rychlost rovně*, jak bylo zmíněno při popisu komponenty v 4.3.4, znamená průjezd mezi prvním a druhým výstupním bodem. *Rychlost*

odbočující pak průjezd mezi prvním a třetím výstupním bodem. Po zvolení rychlosti potvrdíme nastavení stiskem tlačítka *OK*.

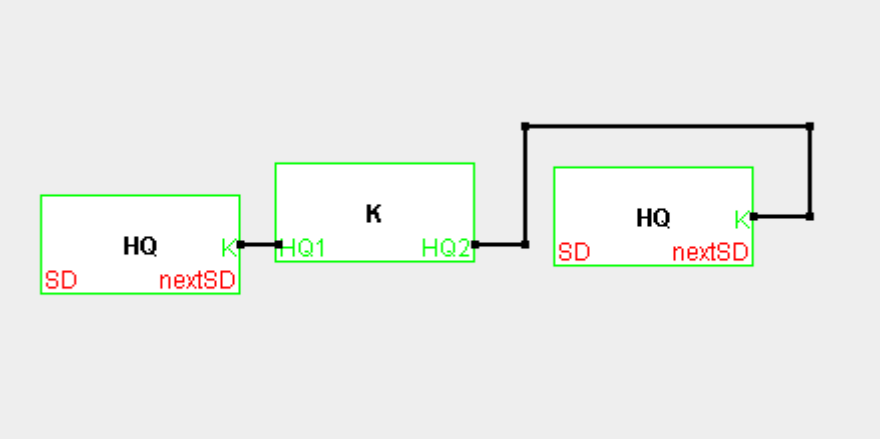
- Pro nastavení komentáře vybereme v kontextovém menu komponenty položku *Komentář*. Potvrzení opět provádíme stiskem tlačítka *OK*.

6.1.3 Tvorba spojů

Další postup, který budeme potřebovat, je spojování komponent spoji. To se děje opět pomocí nástrojové lišty. Vybereme položku *Spoj* a můžeme spojovat komponenty. Podobně jako u vytváření komponent se dostáváme do módu vytváření, tentokrát však spojů. Tento mód má vlastnosti módu vytváření komponent, ale přidává ještě mezistav při nedokončené spojnici. Spoj založíme stiskem levého tlačítka myši na zvolené komponentě. Pokud už do komponenty nelze přidávat další spoje, je na to uživatel upozorněn ve stavové liště a stav editoru zůstává ve módu vytváření spojů. Pokud úspěšně založíme první bod spoje na komponentě, můžeme pokračovat přidáváním jeho zlomových bodů. To se děje jakýmkoliv opět stiskem levého tlačítka myši, tentokrát však do pracovní plochy mimo komponenty. Při prvním zásahu komponenty se spustí proces dokončování spoje -provede se kontrola vazeb a v případě, že alespoň jedna z komponent nepodporuje daný typ spojení, tj. není na ní volná brána daného typu, je spoj zrušen a obnoven stav vytváření spojů.

Příklad: Spojování komponent si představíme na novém diagramu, kde spojíme dvě komponenty typu HQ s komponentou typu K. Na pracovní plochu umístíme dvě komponenty typu HQ a jednu typu K. Z nástrojové lišty vybereme položku *Spoj*. Tentokrát si můžeme všimnout změny kurzoru nad pracovní plochou na vzhled zaměřovacího kříže. Ten značí výběr tvorby spojů. Stiskneme levé tlačítko myši s kurzorem nad námi předem vybranou komponentou typu K. Tím založíme počáteční bod spoje na této komponentě. Dále můžeme pokračovat neomezeným počtem zlomových bodů. Ty vytvoříme stiskem levého tlačítka myši kdekoliv na pracovní ploše, mimo pozic s komponentami. Prvním stiskem levého tlačítka myši nad komponentou se provede ukončení tvorby spojení. To chceme ukončit nad komponentou typu HQ. Na pracovní ploše bychom měli mít dvě. Jednu z nich vybereme a pro dokončení spoje nad ní stiskneme levé tlačítko myši. Po dokončení tvorby spojení se nám otevře dialogové okno s výběrem bran ke kterým se dá spojení připojit. První komponenta K nabízí možnosti *HQ1* a *HQ2*. Druhá komponenta bude nabízet pouze jednu možnost typu *K*. Stiskem tlačítka *OK* potvrdíme vybrané brány. Nyní by se vytvořené spojení mělo vybarvit zeleně na jejich přiřazených branách. Na komponentě typu HQ bude svítit zeleně brána typu K, na komponentě typu K pak brána typu HQ1. Nyní obdobným způsobem vytvoříme spojení mezi zbývající komponentou typu HQ a komponentou typu K. Po dokončení

spoje se již nezobrazí dialogové okno, protože máme na výběr pouze jednu bránu u každé ze spojovaných komponent. Spoj se automaticky připojí k posledním volným branám. Dále si také můžeme všimnout, že se okraj komponenty typu K vybarví zeleně. To znamená, že všechny nutné brány byly obsazeny a komponenta je připravena ke generování.



Obrázek 6.2: Příklad: spojování

Po dokončení stavu vytváření spojů jedním z výše uvedených způsobů lze spoje modifikovat. Kromě posuvu zlomových bodů lze přemisťovat i koncové body a tím měnit logiku spojů. Pokud spoj chytneme za koncový bod, přemístíme ho na jinou komponentu a uvolníme, provedeme pokus o novou vazbu mezi těmito dvěma komponentami. Pokud se vytvoření vazby neprovede, například kvůli obsazenosti bran, je spoj uveden do výchozího stavu (je provedeno znovupřipojení spoje k původním komponentám).

Další modifikací spoje je vytvoření nových zlomových bodů. Nové zlomy je možné vytvořit přidržením klávesy *<Control>* a stiskem levého tlačítka myši zásahovém prostoru spoje. Případně je možné použít kombinaci dvojitého stisku levého tlačítka myši v zásahovém prostoru spoje. Stejnou metodou lze i odebírat existující zlomy. Pouze kursor musí mířit na zlom a ne na spojnici.

Koncovými body spojů i zlomy se dá posouvat. Princip je shodný s posuvem komponent popsaným v předchozí sekci. U koncových bodů spojů má přesouvání smysl pouze jako změna topologie zabezpečovacího zařízení. Tudíž můžeme přesunovat koncové spoje pouze na jiné komponenty. Po přesunutí na jinou komponentu se provede proces připojování. Pokud by bylo na výběr více výstupních bran, pak jsou uživateli nabídnuty v dialogovém okně, jako v případě tvorby spoje. Pokud se však proces spojování nepodaří dokončit, je koncový bod spoje vrácen zpět na původní počáteční komponentu a připojen k původní výstupní bráně.

Spoj, stejně jako komponenta, se dá z diagramu smazat. To se provede vyvoláním kontex-

tového menu kdekoliv v zásahové ploše spoje. Jedinou položkou tohoto kontextového menu je právě akce *Smaž spoj*.

6.2 Práce s vytvořenou topologií

Ve chvíli, kdy máme rozpracovanou, či již dokončenou topologii staničního zabezpečovacího zařízení, je vhodné využít zbylé možnosti editoru. Jak bylo napsáno v úvodu kapitoly, volbou položky *Nový* menu *Soubor* můžeme zrušit rozpracovaný stav editoru[2] a začít tvorbu nového diagramu. Pokud se však uživatel chce k již vytvořenému diagramu dodatečně vracet, bude potřebovat aktuální stav editoru uložit. K tomu lze použít volbu *Uložit jako...*. Ta umožňuje uložit stav editoru na disk. Editor standardně navrhuje název pro uložení souboru *Unnamed.ed*, kde *.ed* je definovaná koncovka pro soubory tohoto editoru.

Stav editoru jednou uložený pomocí volby *Uložit jako...* lze následně vyvolat pomocí volby *Otevřít*. Ta nahradí kompletní stav diagramu v editoru uloženým stavem. Starý stav je zrušen, jako kdyby byla použita volba *Nový*. Pokud se operace nezdaří, například nahráváním nesprávného souboru, je uživatel na tuto skutečnost upozorněn dialogovým oknem. Příkladem může být pokus o nahrání nesprávného souboru.

Dokončený diagram s topologií zabezpečovacích zařízení lze exportovat volbou položky *Exportovat* do výstupního souboru typu XML. Diagram připravený k exportování lze identifikovat dle zeleného vybarvení okrajů všech komponent, které se na něm nacházejí. Pokud uživatel spustí proces exportu pro diagram, který není dokončený, bude na tuto skutečnost upozorněn dialogovým oknem.

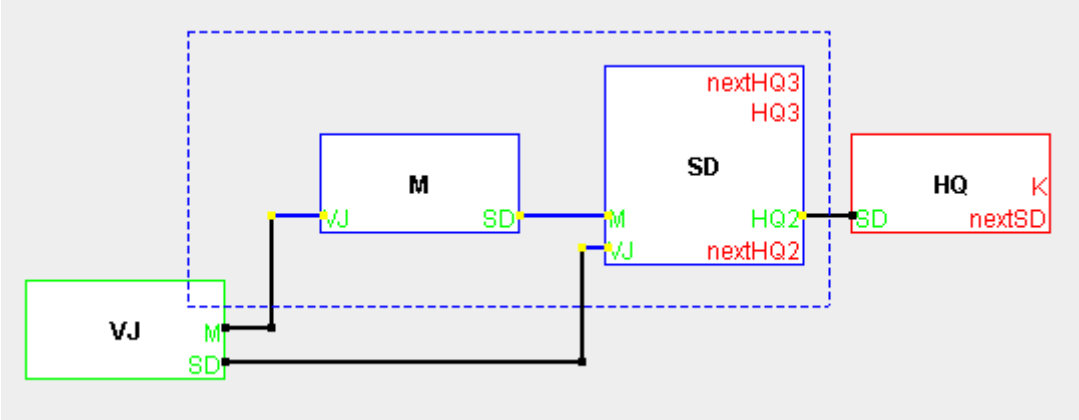
6.3 Další funkce editoru

Editor zpříjemňuje uživatelské rozhraní funkcemi nepřímo souvisejícími s účelem programu. Tyto rozšiřující funkce (podobné s funkcemi v dalších grafických editorech) mají zjednodušit práci s navrhovaným diagramem.

6.3.1 Skupinový výběr

Editor umožňuje skupinový výběr komponent a spojů. Výběr se provede stiskem a tažením levého tlačítka myši na pracovní ploše. V průběhu tažení se zobrazuje obdélník, značící hranici vybírané plochy. Při uvolnění tlačítka myši dojde k výběru komponent zasažených

[2] Stavem editoru se myslí pozice a vlastnosti komponent diagramu, pozice jednotlivých bodů spojů a jim přiřazené brány.



Obrázek 6.3: Příklad: skupinový výběr

výběrovým obdélníkem. Aby byla komponenta vybrána, musí být celá obsažena v obdélníku výběru. Spolu s komponentou se vyberou zároveň i body na ní připojené. Pokud není komponenta vybrána, pak se body na ní připojené nevyberou. Zlomy spojů se vyberou stejným principem, jako u komponenty, t.j. pouze pokud se nacházejí uvnitř výběrového obdélníku. Vybrané komponenty jsou zvýrazněny modrým vykreslením okrajů, které nahradí standardní zobrazení vnitřního stavu zelenou či červenou barvou. Výběr zlomu nebo okrajového bodu spoje je znázorněn žlutým vybarvením tohoto bodu. Pokud jsou dva sousedící body spoje vybrány, pak se čára mezi nimi vybarví modře.

Vybranou množinou komponent a zlomových bodů jako celkem lze posunovat stejným způsobem, který používáme i u posuvu samostatné komponenty nebo zlomového bodu. Při přesunu jsou přesunovány všechny ostatní vybrané prvky (komponenty i zlomy).

Aktuálně vybranou skupinu prvků lze hromadně smazat. Mazání skupiny prvků se provede v kontextovém menu výběru. Kontextové menu otevřeme pravým tlačítkem na jakékoliv komponentě nebo zlomu této množiny. Jedinou položkou menu je právě *Smazat vybrané*.

Ke zrušení aktivního výběru objektů dochází při akcích nesouvisejících s prací s aktuálně vybranou množinou prvků. Zrušení se tedy provádí při jakémkoliv stisku levého či pravého tlačítka myši mimo vybrané komponenty či zlomy nebo při stisku klávesy *<Escape>*.

6.3.2 Historie stavů

Editor si interně ukládá dvacet posledních dokončených stavů diagramu. Ve chvíli, kdy úspěšně dokončíme některou z vybraných akcí, je stav editoru uložen do operační paměti. Mezi vybrané akce patří:

- vytvoření nové komponenty
- vytvoření nového spoje
- posun komponenty
- posun spoje
- vytvoření nového zlomu spoje
- nastavení komentáře komponenty
- nastavení průjezdové rychlosti u komponenty typu SD
- smazání komponenty
- smazání spoje
- zrušení zlomu
- smazání skupiny grafických objektů.

Interně uložené stavy je možné procházet zpět i vpřed pomocí tlačítek nástrojové lišty *Zpět* a *Vpřed*. Existenci minimálně jednoho předchozího uloženého stavu v paměti naznačuje aktivní tlačítko *Zpět* v nástrojové liště. Aktivní tlačítko *Vpřed* ukazuje existenci minimálně jednoho následujícího uloženého stavu v paměti proti stavu aktuálně zobrazeném na pracovní ploše. Pokud je v diagramu zobrazen jiný stav z historie, než je poslední stav, a provedeme jakoukoli z vybraných akcí, které jsem popisoval v předchozím odstavci, všechny následující stavy se z operační paměti smažou a na poslední místo se uloží aktuální stav.

Kapitola 7

Závěr

Cílem této bakalářské práce byla implementace grafického editoru topologie staničního zabezpečovacího zařízení. Tento editor je zjednodušením vytváření vstupního souboru formátu XML do generátoru z diplomové práce[9].

Výsledkem práce je vytvoření prostředku pro tvorbu korektního výstupního XML souboru generátor a zároveň přidává uživatelskou přívětivost a přehlednost grafického editoru. K vizualizaci topologie staničního zabezpečovacího zařízení se využívá grafického rozložení prvků do diagramu. S tímto diagramem může uživatel jednoduše pracovat. Lze měnit topologii, přidávat nové prvky, ukládat a načítat stavy diagramu. Pro zpříjemnění práce byla přidána funkce skupinového pohybu grafických objektů.

Rozšíření je možné úpravou zdrojových souborů editoru. Možným pokračováním by mohla být implementace dynamického vytváření nových komponent zabezpečovacího zařízení. Uživatel by si zvolil výstupní body a jejich brány a navrhl strukturu, kterou má komponenta generovat. Poté by komponentu mohl používat v editoru.

Dalším rozšířením by mohlo být v zvýšení uživatelské přívětivosti. Například, po vzoru Enterprise Architectu[5], by editor mohl umožňovat vytvářet plovoucí popisky spojů a komponent, které by se dynamicky umisťovaly vůči objektu.

Vývoj editoru mi umožnil rozšířit znalosti a postupy pro tvorbu grafického uživatelského rozhraní. Seznámil jsem se s různými aspekty vývoje pod rozšířením Swing programovacího jazyka Java.

Literatura

[1] Abstract Window Toolkit.
http://en.wikipedia.org/wiki/Abstract_Window_Toolkit, stav z 1. 3. 2011.

[2] Qt Jambi.
http://qt-jambi.org/, stav z 1. 3. 2011.

[3] SWT: The Standard Widget Toolkit.
http://www.eclipse.org/swt/, stav z 1. 3. 2011.

[4] Agilní vývoj: Úvod - Zdroják.
http://zdrojak.root.cz/clanky/agilni-vyvoj-uvod/, stav z 1. 3. 2011.

[5] UML tools for software development and modelling - Enterprise Architect UML modeling tool.
http://www.sparxsystems.com/, stav z 1. 4. 2011.

[6] Download free Java software.
www.java.com/getjava/, stav z 1. 3. 2011.

[7] The Jawa Swing tutorial.
http://www.zetcode.com/tutorials/javaswingtutorial/, stav z 1. 3. 2011.

[8] XP-DEV: Subversion, Git & Mercurial Hosting, Project Tracking, Trac Hosting & Agile Tools.
http://www.xp-dev.com/, stav z 1. 3. 2011.

[9] VÍT, P. Nástroj pro generování zabezpečovacího zařízení pro železnici. Master's thesis, České vysoké učení technické v Praze Fakulta elektrotechnická, 2010.

[10] ZATŘEPÁLEK, M. Zabezpečovací začízení pro železniční stanici založené na FPGA. Master's thesis, České vysoké učení technické v Praze Fakulta elektrotechnická, 2008.

Příloha A

Seznam použitých zkratek

FPGA Field-programmable gate array

XML Extensible Markup Language

XSD XML Schema Document

SWT Standard Widget Toolkit

AWT Abstrat Window Toolkit

SVN Apache Subversion

CVS Concurrent Versions System

CD Compact Disc

JRE Java runtime environment

Příloha B

Instalační příručka

Editor není nutno instalovat, je dodáván jako přímo spustitelný soubor. Pro běh editoru je však nutné mít nainstalované prostředí pro běh programů v jazyce Java (Java Runtime Environment, JRE). JRE je možné zdarma stáhnout na stránkách výrobce[6]. Po úspěšné instalaci JRE stačí spustit soubor `editor.jar` příkazem

```
java -jar editor.jar
```

Pokud tento postup selže, pak není správně nainstalován nebo nakonfigurován JRE.

Příloha C

Obsah přiloženého CD

- README.TXT - soubor se základními informacemi
- editor\ - složka s aplikací
  - Editor.jar - soubor aplikace
  - src\ - složka s projektem vývojového prostředí Netbeans
    - dist\ - složka obsahující automaticky vygenerovaný Javadoc
    - src\ - složka obsahující zdrojové soubory editoru
    - další složky a soubory vývojového prostředí
- test\ - složka obsahující testovaná nádraží
  - nadrazi1\ - složka obsahující data nádraží s identifikátorem nadrazi1
  - nadrazi2\ - složka obsahující data nádraží s identifikátorem nadrazi2
  - nadrazi3\ - složka obsahující data nádraží s identifikátorem nadrazi3
  - nadrazi4\ - složka obsahující data nádraží s identifikátorem nadrazi4
- text\ - složka obsahující text bakalářské práce
  - Kulovany-Bakalarska-Prace.pdf - soubor textu bakalářské práce ve formátu pdf
  - LaTeX\ - složka obsahující zdrojové soubory textu práce v systému LaTeX