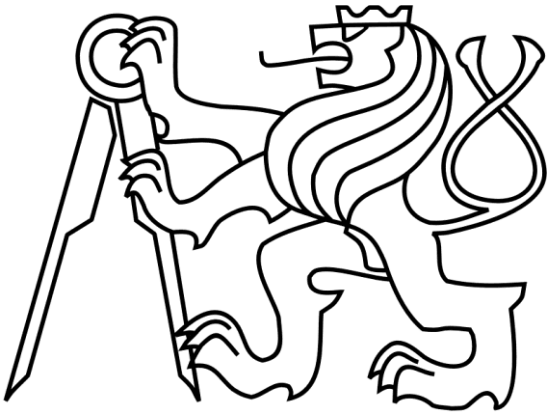
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ



BAKALÁŘSKÁ PRÁCE

APLIKACE PRO SPRÁVU DIAGRAMŮ KRESLENÝCH A ROZPOZNÁVANÝCH NA DOTYKOVÝCH ZAŘÍZENÍCH

Autor: Eliška Roubalová

Program: Otevřená informatika

Obor: Softwarové systémy

Vedoucí práce: Ing. Martin Bresler

15. 5. 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Eliška Roubalová**

Studijní program: Otevřená informatika
Obor: Softwarové systémy

Název tématu: **Aplikace pro správu diagramů kreslených a rozpoznávaných na dotykových zařízeních**

Pokyny pro vypracování:

Navrhněte a implementujte aplikaci, která bude podporovat vizualizaci, modifikaci a reprezentaci ručně kreslených diagramů (vývojové diagramy, konečné automaty). K automatickému rozpoznávání diagramů bude poskytnuta knihovna v C#.

Dílčí kroky:

1. Analyzujte možnosti dotykových zařízení a navrhněte platformu s potenciálním zájmem uživatelů.
2. Navrhněte workflow aplikace pro kreslení, rozpoznávání a vizualizaci diagramů včetně uživatelsky přívětivého grafického rozhraní (GUI). Důraz je kladen zejména na prolnutí kreslení a vizualizace. Zde je důležitá zpětná vazba uživatele na výsledek rozpoznávání a možnost korekce jak vstupu, tak výstupu.
3. K implementaci je doporučeno používat programovací jazyk C#. Výsledná aplikace může mít charakter stand-alone desktopové aplikace, pluginu do aplikace umožňující návrh diagramů (např. MS Powerpoint nebo MS Visio), mobilní nebo webové aplikace.
4. K výsledné aplikaci vytvořte uživatelskou příručku a programátorskou dokumentaci.

Seznam odborné literatury:

[1] Rob Miles, C# Programming, Edition 5.1, August 2014.
[2] Martin Bresler, Truyen Van Phan, Daniel Průša, Masaki Nakagawa, Václav Hlaváč, Recognition System for On-line Sketched Diagrams, ICFHR 2014.

Vedoucí: Ing. Martin Bresler

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26. 3. 2015

PODĚKOVÁNÍ

Ráda bych poděkovala svému vedoucímu práce Ing. Martinu Breslerovi a svému vedoucímu softwarového projektu RNDr. Danielu Průšovi, PhD. za jejich cenné rady, postřehy a názory. Také bych chtěla poděkovat své rodinně a přátelům za jejich trpělivost a podporu.

PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 15. 5. 2015 .....

Abstrakt

Práce se zabývá návrhem a implementací grafického uživatelského rozhraní pro knihovnu rozpoznávače rukou kreslených konečných automatů a vývojových diagramů. Podobné aplikace zatím nejsou příliš rozšířené, pro vývojové diagramy nejsou dostupné žádné komerční rozpoznávače a pro konečné automaty vůbec žádné rozpoznávače, které by obsahovaly grafické uživatelské rozhraní. Cílem je tedy vytvořit přehledné a jednoduché grafické rozhraní, které provede uživatele všemi fázemi rozpoznávání. Návrh grafického rozhraní se zejména zaměřuje na možnost dotykového ovládání a zpětnou vazbu pro uživatele. Aplikace by měla poskytovat uživateli možnost přiměřené korekce výsledku rozpoznání i vizualizace. Výsledný diagram se dá exportovat do PNG souboru.

Klíčová slova

Grafické uživatelské rozhraní, rozpoznávač, konečný automat, vývojový diagram, dotykové ovládání, knihovna, vizualizace.

Abstract

This thesis deals with design and implementation of graphical user interface for a recognizer of hand drawn final automats and flow charts. Applications of this type are not common yet. There are no commercial apps for flow charts and absolutely no apps for final automats which would contain graphical user interface. Therefore, the goal of this thesis is to create clear and simple graphical user interface to guide the user through all recognition phases. Design is mainly focused on the possibility of touch control and feedback for the user. The application should provide the user with reasonable possibilities to correct the recognition result and its visualization. Finished visualized diagram can be exported to the PNG file.

Key words

Graphical user interface, recognizer, final automat, flow chart, touch control, library, visualization.

OBSAH

1 Úvod

Rozpoznávače textu jsou dnes již běžnou součástí mobilních zařízení. Novou metou je tak rozpoznávání složitějších a strukturovaných vstupů, často specifických, a využití těchto rozpoznávačů v dalších praktických aplikacích. Dobrým příkladem takového vstupu jsou matematické vzorce nebo různé typy diagramů a schémat. Úspěch v tomto směru představuje především rozpoznávač matematických vzorců (1), rozpoznávač chemických vzorců (2) nebo rozpoznávač schémat elektrických obvodů (3) (4). V této práci se budu věnovat výhradě diagramům (konkrétně vývojovým diagramům a konečným automatům), které jsou velmi častým grafickým vyjádřením a zároveň mají relativně jednoduchou strukturu. I pro ně existují úspěšné rozpoznávače ((5), (6), (7)).

Vytváření diagramů prostým kreslením je zřejmě nejběžnější a pro mnoho lidí i nepohodlnější způsob. Často je pak potřeba pracně vytvořený diagram přenést do počítače pro další případné úpravy a použití. Různé rozpoznávače tento problém více či méně úspěšně řeší. Alternativní nástroje pro vytváření diagramů pomocí výběru z připravených tvarů jsou mnohdy dostupné pouze v online verzích a nejsou tak intuitivní, jako obyčejné kreslení. Existují dokonce studie, které poukazují na to, že použití takovýchto nástrojů může být pomalejší než přímá kresba (8).

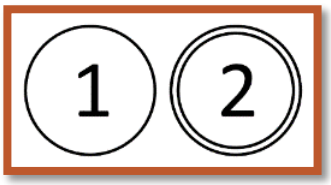
Problém rozpoznávání konečných automatů a vývojových diagramů řeší knihovna poskytnutá jako základ této práce (5). Průzkum ukázal, že existují metody pro rozpoznávání vývojových diagramů, nejsou ale zatím v praxi žádnou komerční aplikací používány. O konečné automaty začal být větší zájem teprve nedávno, pro jejich rozpoznávání tedy existuje jen velice málo metod a také nejsou v praxi žádnými aplikacemi využívány. Neexistují tedy žádné aplikace, které by představovaly grafické uživatelské rozhraní pro tyto rozpoznávače.

1.1 Rozpoznávač konečných automatů a vývojových diagramů

Rozpoznávač se zaměřuje na diagramy s pevnou strukturou, která je poměrně jednoduchá. Skládá se ze symbolů s pevně daným tvarem (uniformních symbolů) a ze šipek propojujících symboly. Dále se v diagramech může vyskytovat text, který popisuje symboly (je uvnitř symbolu), nebo šipky (je v blízkosti šipky). Poskytnutý rozpoznávač přistupuje k rozpoznání jako k optimalizačnímu problému výběru nejvhodnější podskupiny z kandidátů na symboly. Rozděluje proces rozpoznávání na dvě části, nejdřív dojde k detekci kandidátů na jednotlivé symboly a následně k výběru vhodných symbolů, které nejlépe zapadají do celkové struktury diagramu, pomocí strukturní analýzy. Samotná detekce kandidátů se dá také rozdělit na dva podproblémy, na rozpoznání uniformních symbolů a poté na rozpoznání šipek. K rozpoznání textu se přistoupí až na konec, když už je struktura diagramu známá.

U **konečných automatů** se rozlišují následující třídy uniformních symbolů:

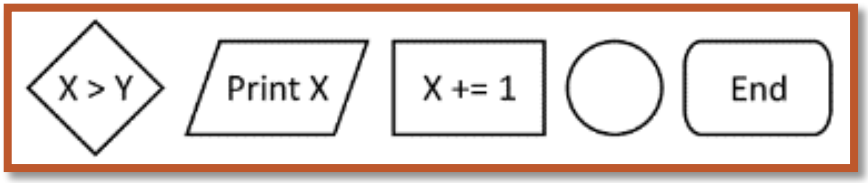
- **Stav** – představuje jeden stav konečného automatu, který může být počáteční, ale není koncový.
- **Koncový stav** – představuje jeden stav konečného automatu, který může být počáteční a musí být koncový.



1. SYMBOL STAVU AUTOMATU A KONCOVÉHO STAVU AUTOMATU.

U **vývojových diagramů** se rozlišují tyto třídy uniformních symbolů:

- **Vstup/výstup dat** – představuje získání dat, nebo jejich zobrazení.
- **Podmínka** – představuje rozhodnutí mezi uzavřenou množinou výsledků, nejběžněji mezi pravdou a nepravdou.
- **Proces** – představuje provedení zadané operace.
- **Spoj** – představuje spojení více procesních toků do jednoho.
- **Terminál** – představuje konec nebo začátek procesního toku.



2. SYMBOLY VÝVOJOVÉHO DIAGRAMU: VSTUP/VÝSTUP, PODMÍNKA, PROCES, SPOJ A TERMINÁL.

Výsledky rozpoznávače jsou velmi dobré, u testovacího souboru diagramů se podařilo správně rozpoznat 93 % symbolů, celkem 55.1 % diagramů bylo rozpoznáno zcela bez chyb (5).

1.2 Cíl práce

Cílem práce je návrh a implementace grafického uživatelského rozhraní pro knihovnu rozpoznávače. Návrh respektuje potřebu dotykového ovládání aplikace, zároveň se snaží o co nejlepší zpětnou vazbu pro uživatele. Současně umožňuje pohodlné ovládání myší v případě, že ovládání dotykem není z nějakého důvodu dostupné.

Samotná implementace využívá jazyk C# (9) a soustředí se na práci se vstupem a vizualizaci rozpoznaného diagramu. Nepatrně rozšířena je i knihovna rozpoznávače, ve které jsou nyní navíc třídy reprezentující jednotlivé diagramy a třída poskytující metody rozpoznání grafickému rozhraní.

2 ANALÝZA

Pro zvolení cílového zařízení a jeho platformy je důležité pokusit se určit cílovou skupinu uživatelů, pro které bude aplikace určena. Také je potřeba zvážit možnosti zadávání vstupu do aplikace na jednotlivých zařízeních zejména s ohledem na typ a velikost jejich displejů. Teprve poté bude možné určit cílovou platformu s ohledem na použitá zařízení a uživatelskou zkušenost.

2.1 CÍLOVÁ SKUPINA

Rozpoznávání konečných automatů a vývojových diagramů má v poměru k celkové populaci celkem úzkou cílovou skupinu uživatelů. Znalost konečných automatů se dá očekávat teprve u vysokoškolsky vzdělaných lidí navíc v technickém směru a matematice. U vývojových diagramů je možné předpokládat širší znalost, používají se například při modelování procesů v některých firmách.

Uživatelé nejsou omezeni věkem ani pohlavím. Primárně je aplikace určena pro české uživatele a má české rozhraní. Umožňuje však poměrně snadnou lokalizaci a po překladu rozhraní už nebudou ani jazyková omezení. Aplikace ovšem vyžaduje vstup od uživatele ve formě nakresleného diagramu, což znemožňuje použití této aplikace nevidomým a motoricky handicapovaným lidem.

Cílovou skupinou jsou tedy zejména vysokoškolsky vzdělaní uživatelé se znalostí procesního modelování a vyšší matematiky, kteří ovládají počítač na uživatelské úrovni, bez rozdílu věku a pohlaví.

2.2 CÍLOVÁ ZAŘÍZENÍ

Vzhledem ke vstupu, který aplikace vyžaduje, je zřejmé, že míří na dotyková zařízení. K pohodlnému kreslení diagramů je však potřeba větší plocha, běžné chytré mobilní telefony nedisponují dostatečnou plochou displeje. Tablety jsou v tomto ohledu lepší, ovšem přesnost kreslení prstem stačí pouze pro malé nekomplikované diagramy, zejména je problém s psaním textu dovnitř jednotlivých symbolů.

Vhodnější je zatím použití stylusu pro vytvoření diagramu a následné ovládání dotykem. Je možné předpokládat, že pro cílovou skupinu bude nejběžnějším zařízením notebook, případně tablet, i s ohledem na potřeby rozpoznávače se tedy jako ideální cílová zařízení nabízejí různé dotykové notebooky, tablety a hybridy mezi notebooky a tablety. V případě nedostupnosti dotykového ovládání je možné aplikaci ovládat i myší, to je ale vhodnější například na úpravy vizualizace diagramu, který byl před tím vytvořen na dotykovém zařízení a uložen.

2.3 Cílová platforma

Výběr cílové platformy silně závisí na cílové skupině uživatelů a na zvoleném zařízení. Knihovna rozpoznávače je vytvořena a optimalizována pro platformu Windows. To sice představuje určité omezení v šíření aplikace, není ale nijak významné. Na hybridních noteboocích a tabletech významnou část trhu pokrývají různé verze systémů Microsoft Windows. Na tuto platformu je také pravděpodobně zvyklá většina uživatelů z cílové skupiny aplikace.

Další plus pro platformu Windows jsou možnosti budoucího rozšiřování aplikace. S ukončením podpory systému Windows XP a blížícím se vydáním nového systému Windows 10 (10) s bezplatným upgradem pro legální uživatele systémů verze 7 a vyšší se dá do budoucna očekávat, že verze 10 zaujme výrazný podíl na trhu hybridních notebooků a tabletů.

Příchod systému Windows ve verzi deset by měl také znamenat příchod takzvaných univerzálních aplikací. Pro všechna zařízení používající systém Windows 10 bude stačit vytvořit pouze jednu aplikaci, která poté bude fungovat na celé škále zařízení od mobilních telefonů, přes tablety a notebooky po klasické stolní počítače. Upravit stávající aplikaci pro Windows bude jistě snazší, než upravovat ji z aplikace pro jinou platformu.

2.4 Výsledek analýzy

Po zvážení všech výše uvedených argumentů je vyvíjená aplikace určena zejména pro hybridní notebooky a tablety s operačním systémem Windows. Je zaměřena na vysokoškolsky vzdělané uživatele, zejména v oblasti matematiky a uživatele používající k návrhům vývojové diagramy. Vyžaduje uživatelskou znalost systému a použití tabletu či notebooku.

3 NÁVRH

Z koncepce rozpoznávače vychází i logické rozdělení práce s aplikací do tří částí: vytvoření vstupního diagramu, případná korekce rozpoznání symbolů a konečná vizualizace s exportem do formátu pro ukládání obrázků. Samotný rozpoznávač totiž přijímá množinu tahů na vstupu a jeho výstupem je seznam rozpoznaných symbolů a vztahů mezi nimi (propojení symbolů navzájem šipkami, příslušnost textu k symbou). Grafické uživatelské rozhraní je tedy také rozděleno do tří na sebe navazujících panelů: panelu pro vytvoření vstupního diagramu, panelu pro drobné opravy rozpoznání a nakonec panelu pro úpravu vizualizace diagramu a export do souboru s obrázkem.

3.1 PANEL PRO ZADÁNÍ VSTUPU

U panelu pro zadání vstupu je zřejmé, že největší plochu musí zaujímat kreslící plátno. Kromě něj je potřeba jen několik tlačítek, která zajišťují přechod mezi fázemi rozpoznávání a upravují vstupní nastavení rozpoznávače. Téměř celou plochu aplikace tedy zaujímá pláto a pod ním se nachází panel s jednotlivými tlačítky.

3. ROZDĚLENÍ VSTUPNÍHO PANELU NA KRESLÍCÍ PLOCHU A OVLÁDACÍ PANEL.

Aby byla uživateli poskytována jasná zpětná vazba o tom, co právě dělá, jsou v levé části panelu umístěna čtyři tlačítka přepínající stav vstupu mezi kreslením a mazáním a mezi zadáváním konečného automatu a vývojového diagramu. Rozpoznávač totiž musí dopředu vědět, který z diagramů rozpoznává. Tato tlačítka zároveň uživatele informují, aktivní volba je vždy zvýrazněna.

4. STAVOVÁ TLAČÍTKA PŘI VÝBĚRU PERA A KONEČNÉHO AUTOMATU

Nejdříve měla tlačítka textový popis, ukázalo se však, že tak zabírají zbytečně mnoho místa a že použití ikon je pro uživatele pohodlnější. S textovými popisky se totiž vždy vešlo pouze jedno ze dvou souvisejících tlačítek. Na panelu tak bylo vždy zobrazeno tlačítko s opačnou volbou, než zrovna uživatel používal, což mohlo vést k jeho zmatení. Popisky tudíž byly nahrazeny výstižnými ikonami.

Pro tlačítka sloužící k výběru pera a gumy jsou symboly běžně používané v grafických editorech, ikony použité v této aplikaci pocházejí z databáze IconFinder (11) a jsou volně použitelné i pro komerční účely. Pro tlačítka přepínající mezi rozpoznáváním konečného automatu a vývojového diagramu běžné symboly nejsou. Tyto ikony byly tedy vytvořeny speciálně pro tuto aplikaci, jako výstižný symbol pro konečný automat byl zvolen běžný stav, pro vývojový automat byl zvolen symbol rozhodnutí. Tyto symboly mají vždy uvnitř ještě zkratku, FA pro konečný automat (final automat / finite automata) a FC pro vývojový diagram (flow chart).

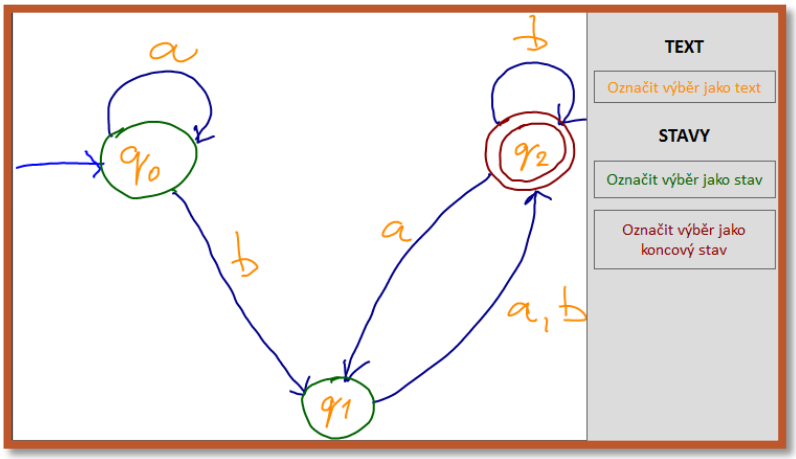
5. KONTRASTNÍ ZVÝRAZNĚNÍ TLAČÍTKA ROZPOZNAT.

Nejdůležitějším tlačítkem je volba rozpoznat, toto tlačítko je tedy umístěno do středu panelu a je kontrastně zvýrazněno oproti ostatním tlačítkům. Ostatní tlačítka mají tmavý popis (ikonu) na světlém pozadí, tlačítko rozpoznat má naopak světlý popis na tmavém pozadí.

V pravé části panelu jsou shromážděna tlačítka pro práci se soubory. Aplikace umožňuje načíst tahy diagramů z XML souborů nebo ze souborů typu InkML. K tomu slouží tlačítko otevřít, které zobrazí běžný dialog pro výběr souboru. Do souborů typu XML umí aplikace nakreslené tahy i ukládat, k tomu slouží tlačítko uložit. Posledním tlačítkem je volba nového vstupu, která vymaže veškeré stávající tahy.

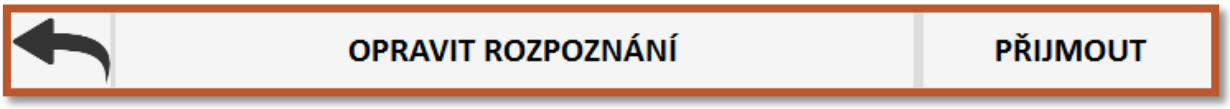
3.2 Panel pro korekci rozpoznání

Tento panel se zobrazí po klepnutí na tlačítko rozpoznat na předchozím panelu. Kreslící plátno tu už zabírá menší plochu, protože bylo potřeba přidat možnosti oprav. To je nakonec řešeno přidáním postranního panelu na pravé straně. Spodní panel s ovládacími tlačítky zůstal velice podobný předchozímu, obsahuje tentokrát tři hlavní tlačítka pro posun mezi fázemi rozpoznávání.



6. PŘIDANÝ POSTRANNÍ PANEL VEDLE KRESLÍCÍ PLOCHY.

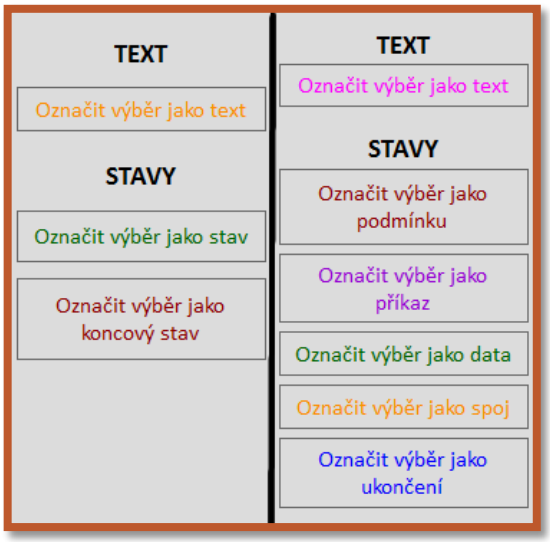
Úplně vlevo se nachází tlačítko zpět, to opět nemá textový popis a využívá běžně zavedené ikony. Vrátí uživatele na předchozí panel, kde je možné vstup překreslit. Kreslící plátno na tomto panelu je totiž výhradně v režimu výběru a není možné tento režim změnit.



7. SPODNÍ OVLÁDACÍ PANEL.

Vpravo je pak tlačítko pro přijmutí rozpoznání, kterým uživatel potvrdí, že jsou symboly rozpoznány správně a je možné přejít k další fázi. Pokud ovšem uživatel provede jakoukoli opravu rozpoznání, nebude moci přejít k vizualizaci, dokud rozpoznání neopraví. K opravě rozpoznání slouží prostřední a největší tlačítko. To spustí nové rozpoznání tahů, tentokrát ale s označenými opravenými symboly.

Zobrazený postranní panel s možnostmi oprav se liší pro konečný automat a pro vývojový diagram. Je to nutné kvůli správným návrhům symbolů i kvůli opravě rozpoznání. Pro výběr správného symbolu se jako první nabízel rozevírací seznam. Na dotykových obrazovkách se ovšem špatně ovládá. Proto bylo nakonec zvoleno řešení v podobě plochých tlačítek, jejichž popisy jsou barevně odlišeny podle příslušných symbolů. To umožňuje jednoduše poskytnout uživateli zpětnou vazbu. Podle barev na tlačítkách jasně vidí, jako které symboly byly tahy označeny. Pokud některým tahům přiřadí při opravě jiný symbol, jejich barva se změní na barvu příslušného nového symbolu.



8. POSTRANNÍ PANEL PRO KONEČNÝ AUTOMAT (VLEVO) A PRO VÝVOJOVÝ DIAGRAM (VPRAVO).

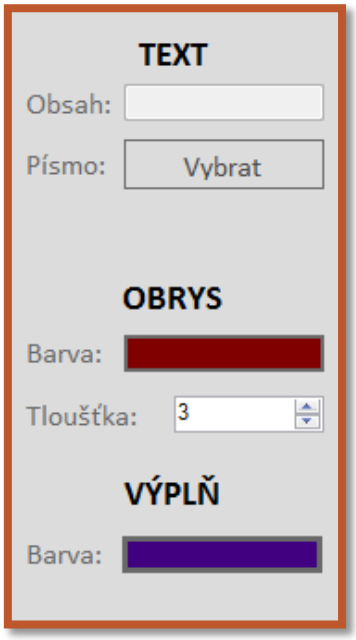
3.3 Panel vizualizace

Po přijmutí rozpoznání na předchozím panelu se provede vizualizace diagramu, která se zobrazí na výstupním plátně tohoto panelu. Výstupní plátno se liší od kreslícího plátna použitého na předchozích dvou panelech, neumožňuje kreslení a je realizováno pomocí bitmapy. Jednotlivé symboly

9. PANEL VIZUALIZACE S VÝSTUPNÍM PLÁTNEM

na plátně se dají vybrat klepnutím myši a posouvat, tím se dají jednoduše upravovat případné nedostatky v rozložení vizualizovaného diagramu.

Vpravo od plátna se opět nachází panel s úpravami. Tentokrát se jedná o úpravy vzhledu jednotlivých symbolů. Po označení symbolu je možné nastavit jeho obrys a výplň (pokud u symbolu dává nastavení výplně smysl), jestliže je vybraný symbol text, dá se upravit jeho obsah a použité písmo, styl, velikost i barva. K výběru vlastností písma i barev jsou použity standardní dialogy systému, na které jsou uživatelé zvyklí i z jiných aplikací pro Windows.



10. POSTRANNÍ PANEL S ÚPRAVAMI VZHLEDU SYMBOLŮ.

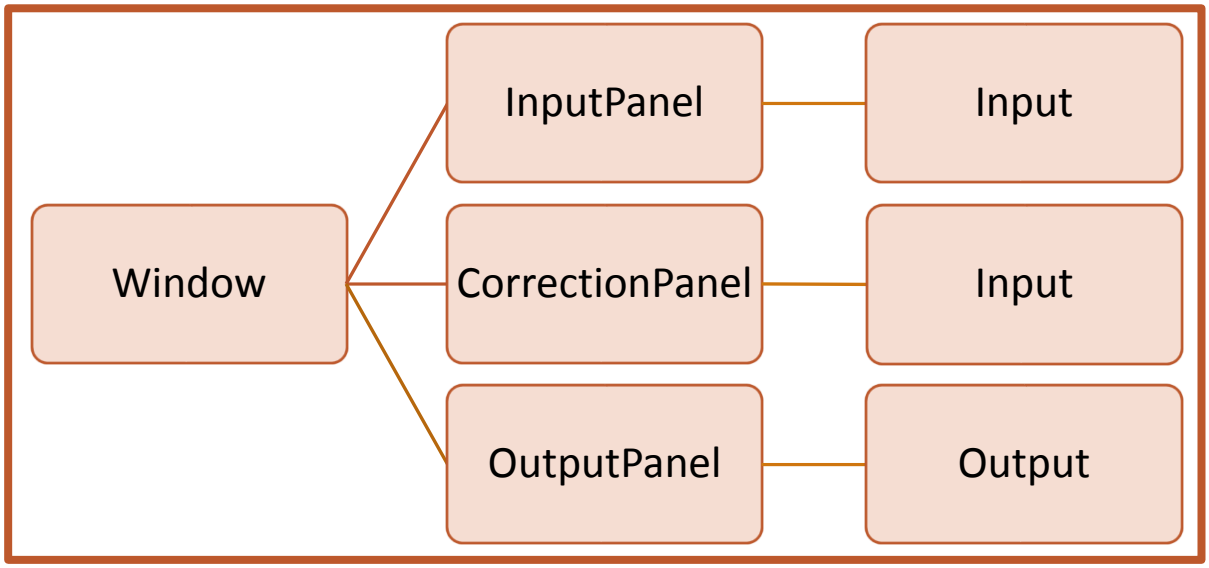
Spodní panel s ovládacími tlačítky opět obsahuje tlačítko zpět, to vrátí proces rozpoznávání na začátek na panel pro získání vstupu. Dále je možné začít úplně nové rozpoznávání s čistým vstupem. V obou těchto případech dojde ke ztrátě rozpoznání i veškerých úprav na panelu vizualizace. Uživatel je na toto chování upozorněn dialogovým oknem.

Posledním tlačítkem, které se na panelu nachází, je tlačítko pro export. To umožní uložit vizualizaci diagramu do souboru jako obrázek. Protože se jedná o předpokládanou konečnou akci, je toto tlačítko opět kontrastní k tlačítkům přecházení mezi fázemi rozpoznání.

4 IMPLEMENTACE

Aplikace je implementována v jazyce C# a byla vytvořena ve vývojovém prostředí Microsoft Visual Studio 2010. Struktura grafického rozhraní je vytvořena z panelů, mezi kterými se přechází v rámci jednoho hlavního okna aplikace. Tyto panely představují jednotlivé části rozpoznávacího procesu a slouží jako kontejnery pro další grafické prvky.

Nejdůležitějšími prvky ve všech panelech jsou plátna. První dva panely obsahují plátno vstupní a poslední panel obsahuje plátno výstupní. Oba typy pláten jsou odvozeny od třídy `UserControl`. Přechod mezi jednotlivými panely zajišťují metody ve třídě `Window`, která tvoří hlavní okno aplikace. Těmito metodami si panely předávají potřebné objekty.



11. STRUKTURA GRAFICKÉHO ROZHRANÍ APLIKACE

4.1 KNIHOVNA ROZPOZNÁVAČE

Jako základ aplikace byla poskytnuta knihovna rozpoznávače. Pro zjednodušení komunikace mezi grafickým uživatelským rozhraním a vlastním rozpoznávačem byly do této knihovny přidány tři nové třídy. První třída obsahuje metody, které rozpoznávač nabízí k použití. Další dvě třídy potom představují jednotlivé druhy diagramů, konečný automat a vývojový diagram.

4.1.1 TŘÍDA RECOGNIZERINSTANCE

Protože inicializace jednotlivých tříd, které rozpoznávač potřebuje ke své funkci, může být časově náročnější a je možné jednou vytvořené instance používat opakovaně, bylo vhodné, aby byl rozpoznávač se všemi potřebnými částmi inicializován pouze jednou při jeho prvním použití. V této třídě byl tudíž použit návrhový vzor singleton, který zajišťuje, že existuje vždy pouze jedna instance dané třídy.

Třída má neveřejný konstruktor a k získání její instance slouží metoda `GetInstance`. Metoda přebírá jako parametr řetězec představující cestu ke spustitelnému souboru aplikace (Tuto cestu vyžaduje metoda pro optimalizaci získaného rozpoznání.). Pokud je tato metoda volána poprvé, je použit neveřejný konstruktor třídy k inicializaci nové instance rozpoznávače se všemi jeho komponentami, jako jsou například klasifikátory symbolů a detektory šipek. Jestliže již existuje vytvořená instance třídy uložená ve statické proměnné `instance`, je vrácena uživateli a nová instance se nevytváří.

Skrz tuto třídu nabízí rozpoznávač dvě veřejné metody, metodu `RecognizeFA` pro rozpoznání konečného automatu a metodu `RecognizeFC` pro rozpoznání vývojového diagramu. Obě metody jsou si velice podobné, rozpoznávač ale pro automaty a diagramy používá odlišné klasifikátory a rozpoznává u nich rozdílné symboly. Bez zvolení typu diagramu uživatelem není možnost zjistit, který diagram je právě rozpoznáván. Proto je podle zvoleného typu diagramu volána z grafického uživatelského rozhraní příslušná metoda.

Každá z těchto dvou metod přebírá jako svůj první parametr prázdnou instanci třídy reprezentující příslušný diagram, tedy instanci třídy `FinalAutomat`, nebo instanci třídy `FlowChart`. V těchto instancích naplní během rozpoznávání potřebné proměnné, se kterými grafické uživatelské rozhraní dále pracuje. Jako druhý parametr přebírají obě metody kolekci tahů `Strokes`. Jedná se o třídu obsaženou v knihovně rozpoznávače, která představuje kolekci jednotlivých tahů, ze kterých se vstupní diagram skládá.

Návratovou hodnotou obou metod je logická hodnota `true`, jestliže rozpoznání proběhlo v pořádku, nebo logická hodnota `false`, pokud se při rozpoznávání vyskytla chyba. Pravděpodobnost výskytu chyby je velice malá, pokud se například rozpoznává špatný druh diagramu (je zvolen typ automat, ale nakreslen je vývojový diagram), je mnohem pravděpodobnější, že výsledkem bude označení všech tahů jako text.

4.1.2 Třídy FinalAutomat a FlowChart

Tyto dvě třídy reprezentují jednotlivé diagramy, třída `FinalAutomat` představuje konečný automat a třída `FlowChart` představuje vývojový diagram. Obě třídy obsahují proměnnou `OriginalStrokes`, ta slouží k uchování původních tahů vstupního diagramu pro možnost návratu v pozdějších fázích rozpoznávání. Rozpoznávač totiž jednotlivé tahy pozměňuje.

Proměnná `AnnotatedSymbols` slouží k opravě rozpoznávání. Ukládají se do ní ty symboly, které uživatel ručně označil. Při opravě rozpoznání je pak kontrolováno, zda nejsou některé tahy označené uživatelem, a pokud ano, rozpoznávač jejich označení nemění a vezme ho v úvahu při určování dalších symbolů. K přidání označeného symbolu slouží metoda `AddAnnotation`.

Ta zkontroluje všechny dosud označené symboly, a pokud je tah v nějakém symbolu označen, odebere tento tah z daného symbolu. Jestliže je však označeným symbolem text, odebere metoda tah z kolekce tahů k označení. Umožňuje tím označit například text uvnitř stavu a následně vybrat celý stav a označit ho, aniž by došlo k přeznačení textu.

Nejdůležitější proměnnou je list rozpoznaných symbolů. Tento list naplní rozpoznávač symboly, které obsahují název symbolu, jemu příslušné tahy a další užitečné informace. S těmito symboly se pak dále pracuje při vizualizaci. Ostatní proměnné a metody jsou odlišné pro každý typ diagramu a slouží ke snazší manipulaci s rozpoznanými tahy.

4.2 VSTUPNÍ PANEL

První panel, který uživatel po spuštění aplikace uvidí, představuje třída `InputPanel`. Kromě proměnných určujících typ rozpoznávaného diagramu obsahuje zejména metody obsluhující událost kliknutí na jednotlivá tlačítka panelu. Z těchto metod stojí za zmínku pouze metody pro otevření souboru, pro uložení nakresleného diagramu do souboru a samotné spuštění rozpoznávání diagramu.

Pro načítání a ukládání diagramů je využit `XmlSerializer`, třída představující kolekci tahů `Strokes` implementuje rozhraní `IXmlSerializable` a umožňuje tak ukládat a načítat tahy do a ze souborů typu XML. Formát XML je jediný, do kterého umožňuje aplikace tahy ukládat, při načítání diagramu ze souboru ale umí zpracovat i formát InkML, potřebné metody poskytuje knihovna rozpoznávače. Otevírání i ukládání souborů využívá standardní dialogová okna systému.

Samotné rozpoznávání tahů se zahájí po klenutí na tlačítko Rozpoznat. Je vytvořena nová kolekce tahů ze seznamu, který vrací metoda `GetStrokes` vstupního plátna. Všechny tahy v kolekcích používaných třídou vstupního plátna jsou instancemi tahů rozpoznávače. Knihovna rozpoznávače definuje vlastní třídu `Stroke` i kolekci `Strokes`, zatímco vstupní plátno zobrazuje tahy z knihovny `Microsoft.Ink`. O vzájemné převádění těchto typů se stará vstupní plátno a panely, které ho používají, pracují pouze s tahy definovanými knihovnou rozpoznávače.

Po získání tahů je podle zvoleného diagramu vytvořena nová instance třídy příslušného diagramu, tedy instance třídy `FinalAutomat`, nebo instance třídy `FlowChart`. Do této instance jsou pro pozdější návrat uloženy získané tahy. Na instanci rozpoznávače vrácené metodou `GetInstance` je zavolána příslušná metoda, které je poskytnuta vytvořená instance diagramu a získané vstupní tahy. Instance diagramu se symboly, které doplnil rozpoznávač, je pak přenesena do dalšího panelu.

4.3 Panel oprav

Na panelu oprav se ve vstupním plátně zobrazí rozpoznaný diagram s barevně odlišenými jednotlivými symboly. I tato třída obsahuje téměř výhradně posluchače událostí pro jednotlivá tlačítka panelu. Výjimkou jsou pouze metody pro zobrazení automatu nebo diagramu, které zobrazí na plátně daný diagram a zajistí zobrazení správných možností oprav.

Tyto opravy jsou realizovány pomocí tlačítek Označit jako… a metody `AddAnnotation` ve třídách diagramů. Každý posluchač tlačítka Označit nejdřív získá vybrané tahy ze vstupního panelu. Ty předá metodě `AddAnnotation` příslušného diagramu spolu s řetězcem představujícím název označovaného symbolu. Tahy, které byly opravdu označeny (nebyly odebrány kvůli označenému symbolu textu), zvýrazní jejich zesílením a obarvením na barvu příslušného symbolu na vstupním plátně.

Jestliže byly provedeny nějaké opravy rozpoznaného diagramu, není možné pokračovat k dalšímu panelu, dokud nedojde k opravě rozpoznání. Oprava rozpoznání probíhá stejně jako rozpoznání původní pouze s tím rozdílem, že rozpoznávač kontroluje seznam označených symbolů diagramu, a pokud není prázdný, zahrne označené symboly do svého procesu.

Pokud nebyly provedeny žádné změny, nebo bylo rozpoznání opraveno, je možné přijmout rozpoznaný diagram a pokračovat k dalšímu panelu. Jak již bylo zmíněno, rozpoznávač jednotlivé tahy během procesu rozpoznávání pozměňuje, pro úspěšnou vizualizaci je ale potřeba znát původní tahy jednotlivých symbolů. Proto před samotným přechodem k panelu vizualizace musí být tahy jednotlivých symbolů upraveny do původní podoby a zároveň jsou pro každý symbol určeny souřadnice horního levého a dolního pravého vrcholu jeho ohraničujícího obdélníku v pixelech.

4.4 Vstupní plátno – třída Input

Oba výše zmíněné panely používají pro získání a korekci vstupu třídu `Input` představující vstupní plátno. Tato třída využívá knihovnu `Microsoft.Ink`, konkrétně její objekt `InkOverlay`, který kontroluje kreslení, mazání a výběr tahů na plátně. Tahy získané objektem `InkOverlay` jsou instancemi třídy `Microsoft.Ink.Stroke`, rozpoznávač ale potřebuje tahy reprezentované jeho vlastní třídou `Recognizer.Stroke`.

Proto obsahuje instance plátna proměnnou `StrokesMap`, což je kolekce typu `Dictionary` sloužící k mapování tahů rozpoznávače `Recognizer.Stroke` na tahy plátna `Ink.Stroke` pomocí ID. K posluchačům událostí přidaného a smazaného tahu na objektu `InkOverlay` jsou za účelem mapování přidány metody převádějící tah `Ink.Stroke` na tah `Recognizer.Stroke` a naopak, které ukládají nebo odebírají příslušnou dvojici

`Ink.Stroke.ID` (klíč) a `Recognizer.Stroke` (hodnota) do nebo z kolekce `StrokesMap`.

Komunikaci plátna s panely grafického uživatelského rozhraní zajišťují metody `ShowStrokes` a `GetStrokes`. Metoda `ShowStrokes` zobrazí kolekci `Recognizer.Strokes` na plátno převedením jednotlivých tahů rozpoznávače na příslušné tahy plátna. Druhá metoda naopak vytvoří kolekci `Recognizer.Strokes` z tahů obsažených ve slovníku `StrokesMap`.

Důležitou metodou pro zobrazení tahů rozpoznávaných diagramů je metoda `PackToInput`. Velikost plátna na vstupním panelu je totiž jiná než velikost plátna na panelu oprav (kvůli postrannímu sloupci), tato metoda zajistí upravení tahů tak, aby se vždy vešli do plochy plátna a zároveň ji využili celou.

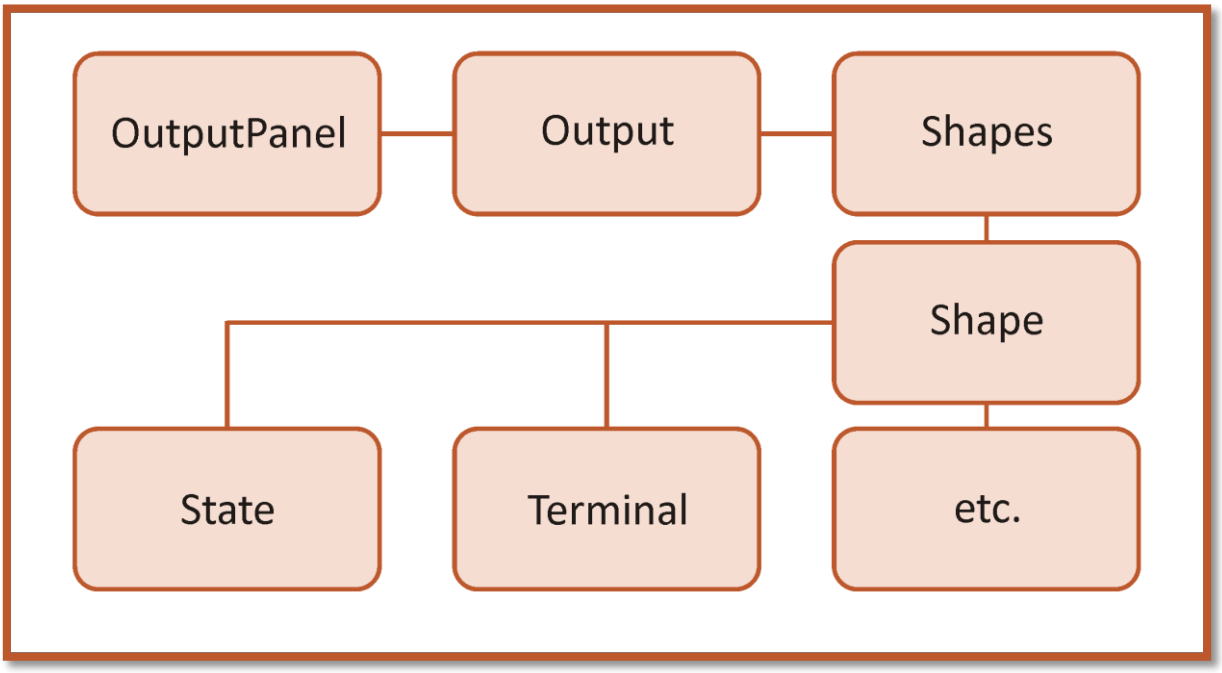
Pro zobrazení rozpoznaného diagramu přidává vstupní plátno metody k obarvení kolekce tahů nebo k úpravě jejich šířky. Tahy předané ve formě seznamu objektů `Recognizer.Stroke` jsou nejdříve vyhledány ve slovníku a převedeny na příslušné tahy `Ink.Stroke`. Obarvit tahy je možné i s pomocí pole indexů odkazujících přímo do seznamu `Ink.Strokes` instance `InkOverlay`.

Protože instance `InkOverlay` představující kreslící plochu má jiné rozlišení než výstupní plátno, bylo nutné implementovat metodu převádějící bod v souřadnicích `InkSpace` na bod se souřadnicemi v pixelech `GetPixelPoint`. Při přepočítávání tahů na velikost plátna je zase potřeba metoda převádějící velikost objektu Input v pixelech na velikost se souřadnicemi v `InkSpace`.

Objekt `InkOverlay` sice sám ošetřuje výběr tahů na plátně, pokud je v editačním módu `Select`, ostatní třídy grafického uživatelského rozhraní ale pracují s tahy rozpoznávače. Vybrané tahy jsou proto vyhledány v mapě a do volajícího místa grafického rozhraní je předán seznam tahů `Recognizer.Stroke`.

4.5 Panel vizualizace

Posledním panelem aplikace je panel vizualizace. Rozpoznané symboly jsou zde převedeny na grafické objekty. Na rozdíl od předchozích panelů neobsahuje panel vizualizace vstupní plátno, ale výstupní plátno s kolekcí tvarů. Obsahuje také posluchače událostí pro tlačítka představující nastavení jednotlivých tvarů.



12. STRUKTURA VÝSTUPNÍHO PANELU.

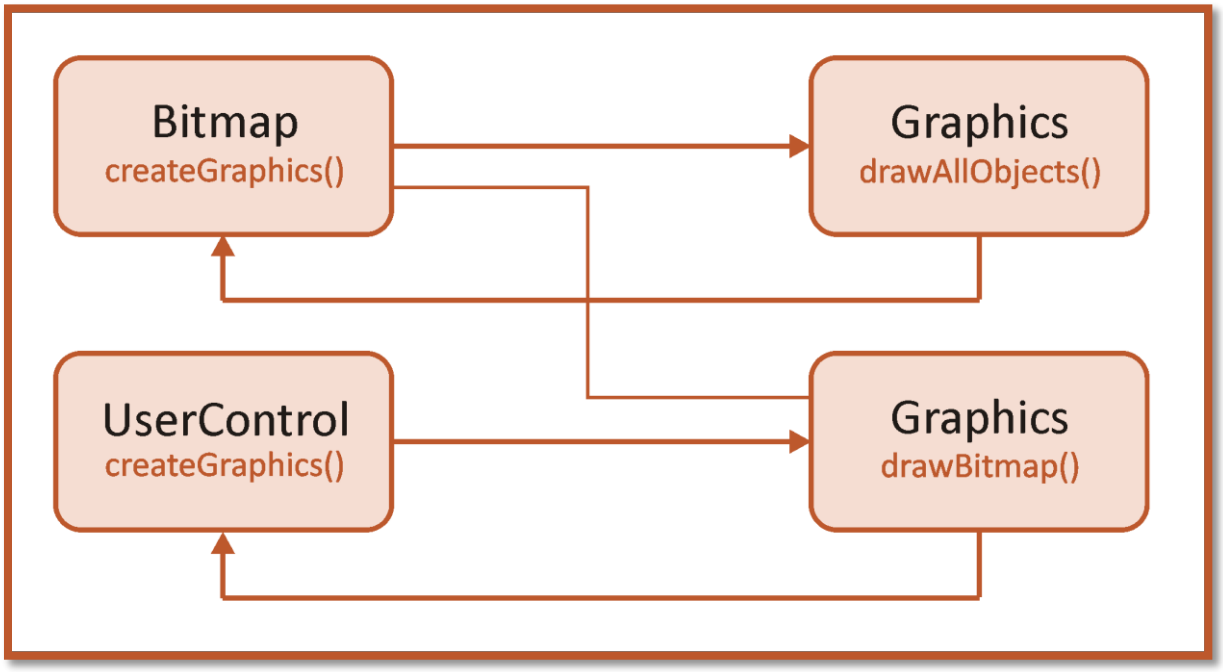
Postranní panel s nastavením vzhledu jednotlivých tvarů vždy umožňuje nastavit jen ty vlastnosti, které dávají u vybraného objektu smysl. Při výběru objektu klepnutím myši je vygenerována událost `ShapeSelected`, kterou výstupní panel zachytí a podle typu vybraného tvaru změní panel s nastavením.

4.6 Výstupní plátno – třída Output

Na rozdíl od vstupního plátna není výstupní plátno realizováno pomocí knihovny `Microsoft.Ink`, ale používá metodu vykreslování objektu `Graphics` do objektu `Bitmap`. Výstupní plátno tak implementuje metodu takzvaného dvojitého bufferování (12).

Pokud se objekty na plátně často mění (posuny, zvětšování a zmenšování), může být vykreslování přímo do okna časově náročné a pohyb nemusí působit dostatečně plynule. Při použití techniky dvojitého bufferování jsou nejdříve objekty vykresleny do obrázku (bitmapy) a teprve potom je vykreslen celý obrázek.

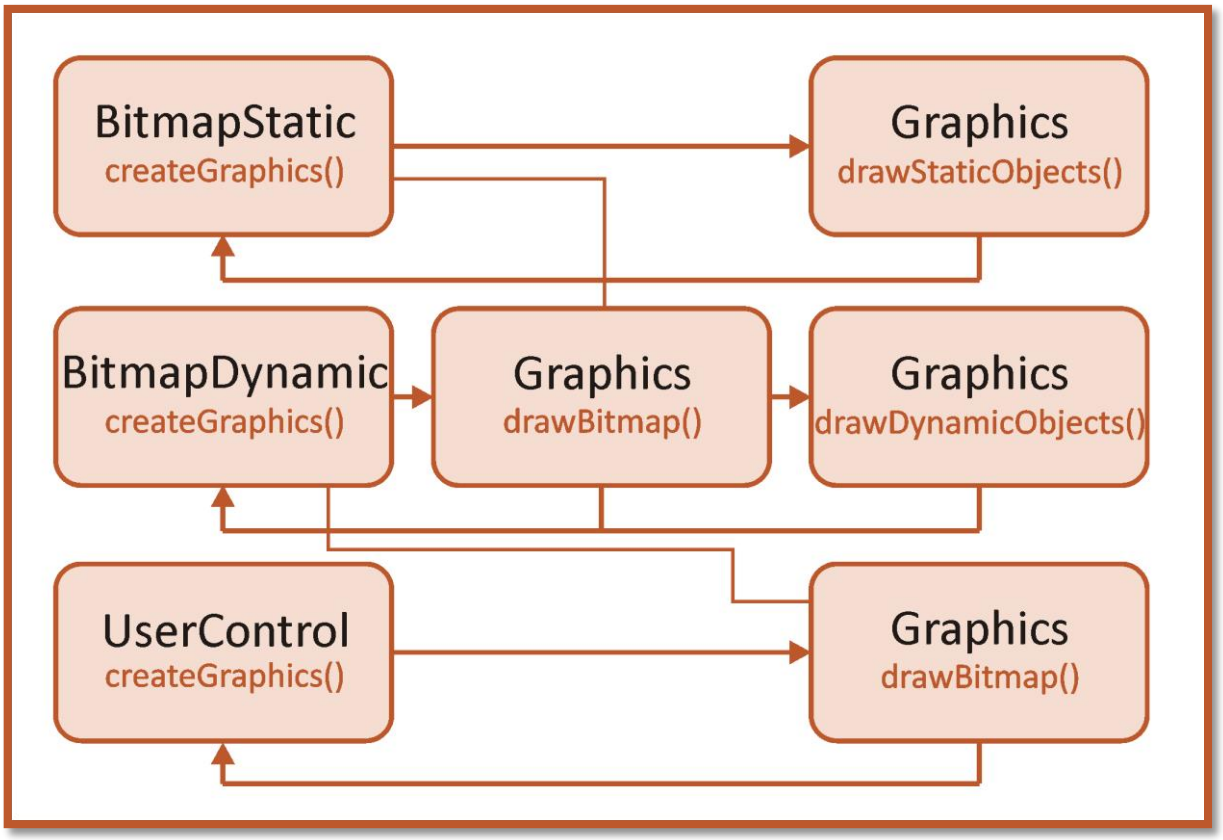
14. BEZ POUŽITÍ DVOJITÉHO BUFFEROVÁNÍ.



13. S POUŽITÍM DVOJITÉHO BUFFEROVÁNÍ.

Výstupní plátno navíc nepoužívá pouze jeden objekt `Bitmap`, ale hned dva takové objekty. Do jedné bitmapy se vykreslují statické grafické objekty, tedy ty objekty, které nejsou zrovna vybrány a tudíž se nemění. Tuto bitmapu tak není potřeba vykreslovat vždy znovu, stačí ji obnovit, jen když se mění všechny objekty na plátně.

Do druhé bitmapy se vykreslují objekty dynamické, tedy ty, které jsou zrovna vybrané a nějakým způsobem se mění. Nejdříve dojde k vykreslení statické bitmapy (pokud je to potřeba), ta je poté vykreslena do bitmapy dynamické a je k ní přidáno vykreslení dynamických objektů. Nakonec je na plátno vykreslena dynamická bitmapa obsahující nyní všechny objekty.



15. POUŽITÍ STATICKÉHO A DYNAMICKÉHO BUFFERU.

Kromě metod pro vykreslování s pomocí dvojitého bufferu obsahuje výstupní plátno zejména posluchače událostí. Důležitými událostmi jsou klepnutí, tažení a zdvih (myši, prstu, stylusu). Ve chvíli kdy dojde ke klepnutí je zkontrolována souřadnice klepnutí. Pokud je na dané souřadnici umístěn vybraný tvar, může začít akce posunu nebo změny velikosti. Jestliže bylo klepnuto do prázdné oblasti na plátně, začne se vytvářet obdélník pro výběr tvaru. Při ovládání myší je také odlišeno klepnutí pravým tlačítkem, které začne akci posunu plátna.

Při tahu je důležitá akce určená při klepnutí. Pokud bylo klepnuto dovnitř do objektu, bude se objekt postupně posouvat. Jestliže bylo klepnuto na některý z manipulátorů objektu, bude se objekt podle příslušného manipulátoru zvětšovat, nebo zmenšovat (podle směru tahu). Při ovládání aplikace myší toto platí pro předchozí klepnutí levým tlačítkem. Tah se stisknutým pravým tlačítkem myši bude postupně posunovat počátek plátna a tudíž i všechny objekty na něm vykreslené.

Po zdvihu je opět kontrolována vybraná akce. Jestliže bylo klepnuto do tvaru, který nebyl vybrán, dojde k jeho vybrání. Pokud se vytvářel výběrový obdélník, dojde k výběru tvaru, který se nachází uvnitř. Na konci posunu nebo změny velikosti tvaru je daná akce ukončena.

Výstupní plátno obsahuje kolekci obsažených tvarů, kterou představuje instance třídy `Shapes`. Zprostředkovává tak možnosti nastavení vlastností tvarů jako jsou barva obrysu nebo

použité písmo panelu vizualizace. Ten díky události vyvolané při výběru tvaru ví, jaká nastavení jsou pro zvolený typ tvaru možná, nepřistupuje ale ke tvaru přímo, používá metody výstupního plátna, které pak provede požadované změny u vybraného tvaru v kolekci `Shapes`.

Export diagramu do souboru typu PNG také zajišťuje výstupní plátno. Vykreslí všechny objekty (statické i dynamické), čímž dojde k jejich aktualizaci v objektech bitmap. Bitmapu představující dynamický buffer pak uloží do vybraného souboru.

4.7 Kolekce tvarů – třída Shapes

Třída `Shapes` obaluje seznam všech tvarů, které výstupní plátno obsahuje. Kromě seznamu všech tvarů uchovává i informaci o tom, který tvar je aktuálně vybraný, a objekt příslušné kolekce manipulátorů. Poskytuje metody pro přidání tvaru do seznamu i pro manipulaci s obsaženými tvary.

Při vykreslování tvarů volá metodu `Draw` na jednotlivých tvarech podle typu vykreslování. Můžou být vykresleny všechny tvary obsažené v seznamu, nebo mohou být vybrány a vykresleny jen označené tvary. Při klepnutí do plochy plátna zkontroluje všechny tvary, vybere ten, který se nachází v místě klepnutí. Pro vybraný tvar pak vytvoří novou kolekci manipulátorů.

4.8 Abstraktní tvar – třída Shape

Jako vzor pro všechny tvary slouží abstraktní třída `Shape`, od které všechny třídy tvarů dědí. Tato třída definuje základní proměnné společné všem tvarům. Tvary jsou na plátně umístěny podle souřadnic levého horního a pravého dolního rohu svého ohraničujícího obdélníku. Tyto hodnoty byly převedeny ze souřadnic InkSpace na pixely při přechodu z panelu oprav k panelu vizualizace.

Dalšími společnými proměnnými jsou především vlastnosti určující vzhled tvarů jako je barva a šířka obrysu, zda je tvar vyplněn barvou a jakou. Logické proměnné `Line` a `Selected` pak určují, po řadě zda je tvar čára (například šipka) a zda je tvar aktuálně vybraný.

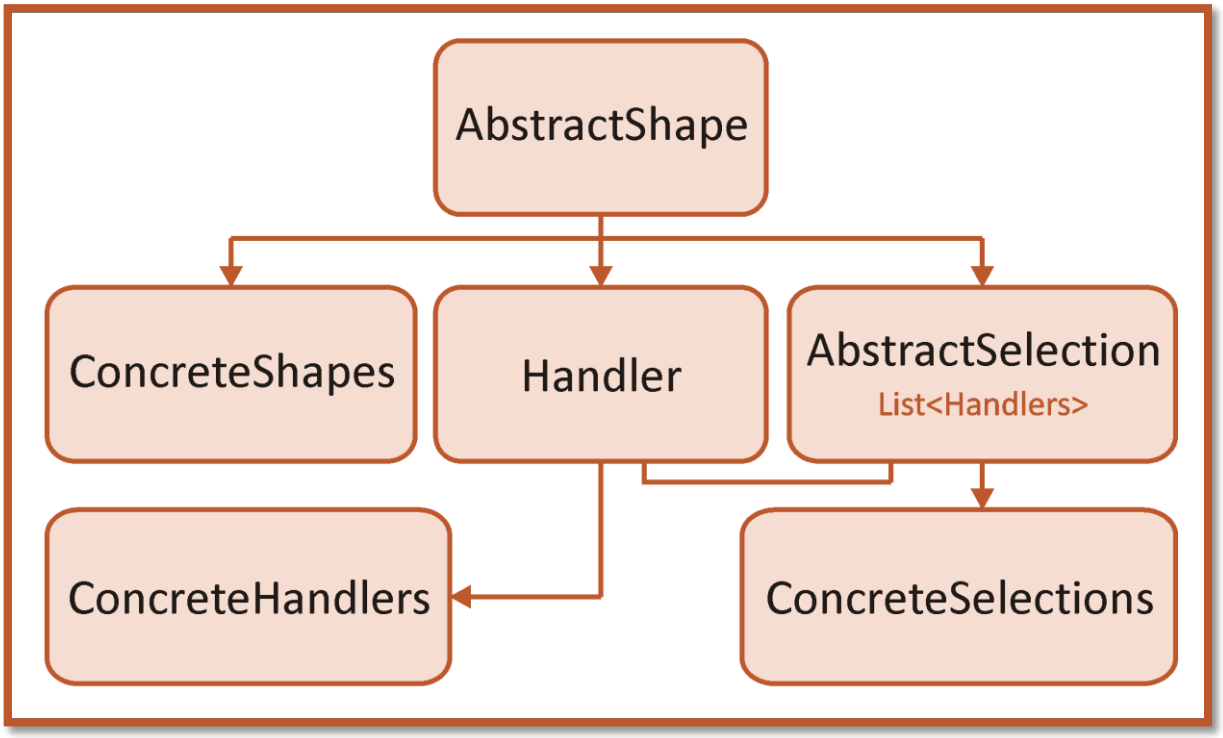
Především ale třída definuje virtuální metodu `Draw`, kterou musí všechny tvary implementovat. Dále definuje základní způsob určování, zda je zadaný bod obsažen v tvaru, a základní manipulaci s tvary při jejich posunu a při změně jejich velikosti. Tyto metody mohou zděděné třídy využít, nebo je mohou překrýt vlastními metodami.

4.9 Manipulátory – třída Handlers

Ve chvíli kdy dojde k výběru nějakého tvaru na plátně, je pro tento tvar vytvořena kolekce manipulátorů. Kolekce manipulátorů je třída dědící od abstraktní třídy `AbstractSelection`. Ta definuje základní vlastnosti kolekce, k jakému tvaru obsažené manipulátory patří a jak se mají chovat při posunu, změně velikosti tvaru nebo vykreslení vybraného objektu. Tato abstraktní třída přitom také dědí od základní třídy `Shape` a má proto své určující body i vlastnosti vzhledu.

Konkrétní implementace třídy `AbstractSelection` při své inicializaci vytvoří nové instance jednotlivých typů manipulátorů, které dávají u tvaru příslušnému kolekci smysl, a přidá je do svého seznamu manipulátorů. Zároveň překryje metodu `Draw` ze třídy `Shape` tak, aby se jednotlivé manipulátory na vybraném objektu správně vykreslovaly.

Jednotlivé manipulátory pak všechny dědí od abstraktní třídy `Handler`, která je také odvozená od základní třídy `Shape`. Základními vlastnostmi manipulátoru je jeho pozice na tvaru, a zda je aktuálně viditelný. Pozice je definována řetězcem podle světové strany (NW, N, NE…). Konkrétní implementace manipulátorů překrývají metodu `Draw` a definují metodu `Reposition`, která zajišťuje, aby při změnách tvaru zůstal manipulátor na správném místě.



16. STRUKTURA DĚDIČNOSTI

5 Závěr

Úvodní analýza problému ukázala, že nejvhodnějšími zařízeními pro tuto aplikaci jsou dotykové notebooky, tablety a hybridy mezi notebooky a tablety. Na tato zařízení byl tedy zaměřen návrh grafického uživatelského rozhraní. Vzhledem k tomu, že zatím nejsou žádné komerční aplikace, které by představovaly grafické uživatelské rozhraní pro rozpoznávače konečných automatů a vývojových diagramů, neexistují žádné osvědčené principy návrhu a průchod rozpoznáváním i návrh designu aplikace je tedy zcela originální.

Při návrhu byl kladen důraz zejména na snadné ovládání aplikace na dotykových zařízeních. Ukázalo se, že není příliš pohodlné kreslit diagramy přímo prsty, protože kresby diagramů vyžadují jistou míru přesnosti, aby mohly být úspěšně rozpoznány, a plocha prstu je zejména pro psaní textu do symbolů příliš veliká. U ostatních grafických ovládacích prvků se toto omezení neprojevilo.

Samotná aplikace je implementována v jazyce C# a využívá poskytnutou knihovnu rozpoznávače. Rozpoznávač neměl žádné komunikační rozhraní, které by umožňovalo jednoduše předat tahy k rozpoznání a získat výsledný rozpoznaný diagram. Proto byly do knihovny rozpoznávače přidány tři třídy reprezentující komunikační bod pro grafické uživatelské rozhraní, diagram konečného automatu a vývojový diagram.

Při práci se vstupem bylo nutné řešit konverze mezi tahy rozpoznávače a tahy vstupního plátna. Za tímto účelem obsahuje vstupní plátno několik metod, které zajišťují mapování odpovídajících tahů mezi sebou a další manipulaci s nimi. Problémem bylo i to, že během rozpoznávání došlo ke změně tahů. Přechod od rozpoznání k vizualizaci totiž vyžaduje znalost původní polohy tahů. Bylo tedy nutné jednotlivé tahy po rozpoznání upravit do jejich původní podoby.

Vizualizace jednotlivých symbolů diagramů je realizována vytvořením instance odpovídajícího grafického objektu podle názvu symbolu. Aby vizualizace rozpoznaného diagramu umožňovala změnu velikosti a polohy symbolů, byly přidány manipulátory pro symbol, který je aktuálně na plátně označený. Vykreslování grafických objektů na výstupní plátno využívá princip dvojitého bufferování, navíc s bitmapou zvlášť pro statické nepozměněné objekty a zvlášť pro dynamické měnící se objekty. Toto řešení umožňuje překreslovat vždy jen ty objekty, se kterými uživatel manipuluje, a zbytek diagramu zbytečně nepřekreslovat.

6 REFERENCE

1. **Alvaro, Francisco, Sánchez, Joan-Andreu a Benedí, José-Miguel.** Recognition of On-line Handwritten Mathematical Expressions Using 2D Stochastic Context-Free Grammars and Hidden Markov Models. *Pattern Recognition Letters.* 2014.

2. **Ouyang, Tom Y. a Davis, Randall.** Recognition of Hand Drawn Chemical Diagrams. *AAAI Conference on Artificial Intlligence.* [Online] 2007. http://people.csail.mit.edu/ouyan/papers/Ouyang2007AAAI.pdf.

3. **Dreijer, Janto F.** Interactive Recognition of Hand drawn Circuit Diagrams. *University of Stellenbosch.* [Online] 2006. http://dip.sun.ac.za/~janto/circuit_sketch.pdf.

4. **Feng, Guihuan, Viard-Gaudin, Christian a Sun, Zhengxing.** On-line hand-drawn electric circuit diagram recognition using 2D dynamic programming. *Pattern Recognition.* 2009, Sv. 42, 12.

5. **Bresler, Martin, a další.** Recognition System for On-line Sketched Diagrams. *ICFHR.* 2014.

6. **Carton, Ceres, Lemaitre, Aurélie a Couasnon, Bertrand.** Fusion of Statistical and Structural Information for Flowchart Recognition. *International Conference on Document Analysis and Recognition (ICDAR).* 2013.

7. **Delaye, Adrien.** Structured prediction models for online sketch recognition. *Unpublished manuscript.* [Online] 2014. https://sites.google.com/site/adriendelaye/home/news/unpublishedmanuscriptavailable.

8. **Miyao, Hidetoshi a Maruyama, Rei.** On-Line Handwritten flowchart Recognition, Beautification and Editing System. *Frontiers in Handwriting Recognition (ICFHR).* 2012.

9. **Miles, Rob.** *C# Programming.* místo neznámé : Department of Computer Science, University of Hull, 2014.

10. **Windows 10. *Microsoft.*** [Online] 2015. http://windows.microsoft.com/cs-cz/windows-10/about.

11. IconFinder. [Online] 2015. https://www.iconfinder.com/.

12. Double Buffering With GDI+. *CodeProject.* [Online] 2002. http://www.codeproject.com/Articles/1819/Double-Buffering-With-GDI?q=double+buffer.

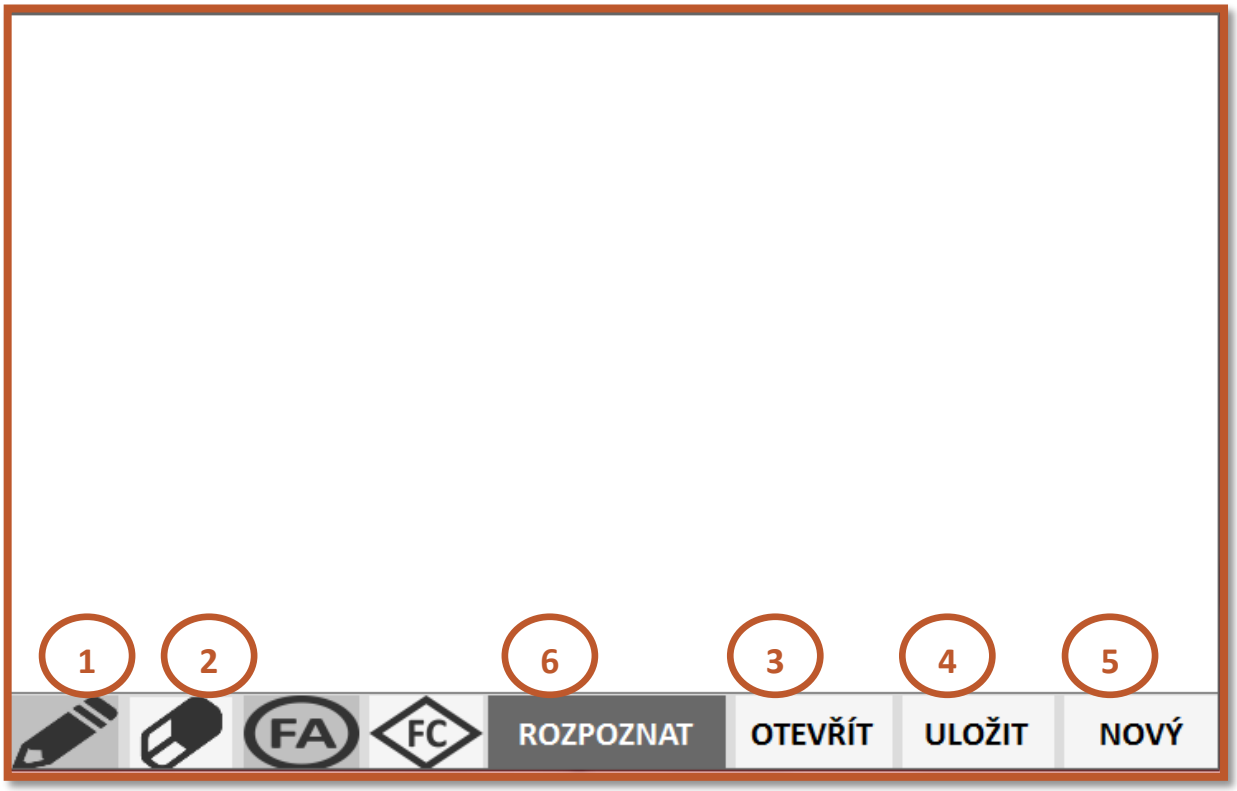
7 Seznam obrázků

PŘÍLOHA A – UŽIVATELSKÝ MANUÁL

Aplikace, jejíž manuál právě čtete, slouží k rozpoznávání ručně kreslených vývojových diagramů a konečných automatů. Její ovládání vyžaduje dotykovou obrazovku, stylus či myš. Pro kreslení diagramů je však nejvhodnější použití stylusu.

KROK 1 – VYTVOŘENÍ VSTUPU

Po spuštění aplikace se zobrazí úvodní obrazovka sloužící k vytvoření diagramu, který bude dále rozpoznán a upraven. Při vytváření diagramu máte na výběr z několika možností:



17. ÚVODNÍ OBRAZOVKA PRO VYTVÁŘENÍ VSTUPNÍHO DIAGRAMU.

a) Použití nástroje **Pero (1)** pro kreslení na plochu plátna, případně použití nástroje **Guma (2)** pro smazání nevhodných tahů. Tímto způsobem sami vytvoříte diagram podle svých představ.
b) Načtení diagramu ze souboru. Pokud máte diagram (například uložený z předchozího používání aplikace apod.) v souboru typu InkML nebo XML, můžete ho klepnutím na tlačítko **Otevřít (3)** načíst do aplikace. Pro výběr souboru se zobrazí standardní dialogové okno Otevřít soubor.

Ve chvíli, kdy je vytvořen nějaký vstupní diagram, je možné ho před pokračováním k rozpoznání **uložit (4)** do souboru ve formátu XML pro pozdější použití. Pro ukládání se opět zobrazí standardní dialogové okno Uložit soubor.

Pokud se stane, že nejste s nakresleným digramem spokojeni a chcete začít znovu, ale nechcete pracně diagram po jednotlivých tazích mazat, klepněte na tlačítko **Nový (5)**. Dojde k vymazání veškerých vložených tahů a nastavení aplikace se vrátí do stavu po spuštění.

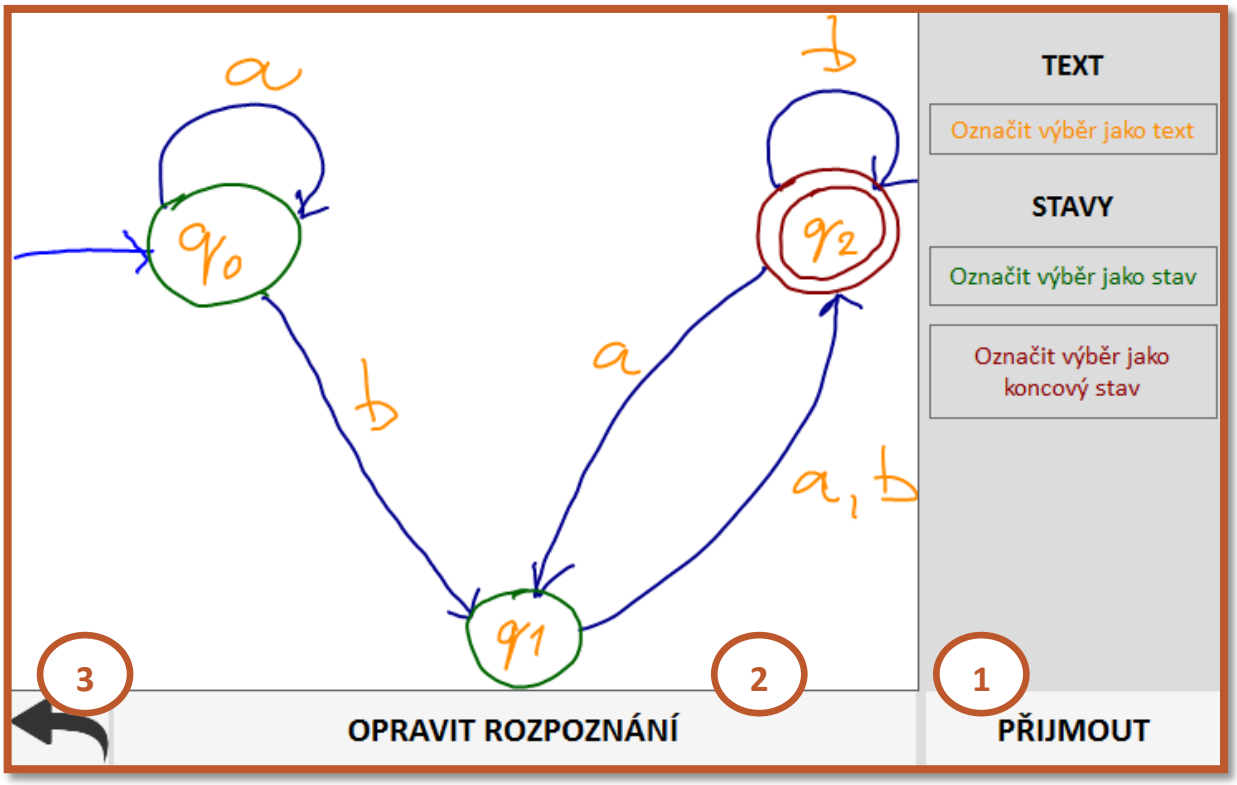
Než budete pokračovat rozpoznáním diagramu, zkontrolujte, že je zvolen správný typ vytvořeného diagramu. Tlačítka pro přepínání mezi konečným automatem a vývojovým diagramem se nacházejí mezi nástrojem Guma a tlačítkem Rozpoznat. Pokud bude zvolen špatný typ diagramu, rozpoznání nebude úspěšné.

18. TLAČÍTKA PRO PŘEPÍNÁNÍ TYPU DIAGRAMU (ZVOLEN KONEČNÝ AUTOMAT).

K dalšímu kroku přejdete klepnutím na tlačítko **Rozpoznat (6)**.

Krok 2 – Korekce rozpoznání

Pokud byl diagram úspěšně rozpoznán, zobrazí se na plátně jednotlivé symboly barevně odlišené podle svého typu. V pravé části přibyde panel pro opravy rozpoznání. Podle výsledku rozpoznání můžete pokračovat následovně:



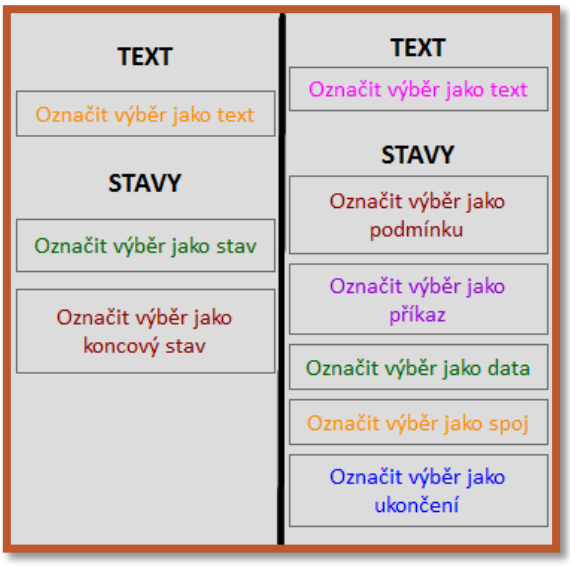
19. ROZPOZNANÝ KONEČNÝ AUTOMAT.

a) Jestliže byl diagram rozpoznán správně a nechcete provádět žádné opravy, pokračujte k dalšímu kroku klepnutím na tlačítko **Přijmout (1)**.

Jestliže jsou v rozpoznání jen drobné nedostatky, označte pečlivě špatně rozpoznané tahy symbolů (zakroužkujte je) a výběrem vhodné možnosti na **panelu oprav** je označte jako požadovaný symbol. Poté klepněte na tlačítko **Opravit rozpoznání (2)** a novu zkontrolujte jednotlivé symboly.

b) Pokud je rozpoznání výrazně špatné, můžete se zkusit klepnutím na tlačítko **Zpět (3)** vrátit ke kroku jedna a překreslit vstupní diagram.

c) Jestliže vidíte diagram stejný jako při jeho vytváření bez barevně označených symbolů, zkuste se také vrátit ke kroku jedna a zkontrolujte, zda byl zvolen správný typ rozpoznávaného diagramu. Při výběru špatného typu totiž buď nedojde k rozpoznání symbolů vůbec, nebo bude rozpoznání výrazně špatné.



20. PANEL OPRAV PRO KONEČNÝ AUTOMAT (VLEVO) A PRO VÝVOJOVÝ DIAGRAM (VPRAVO).

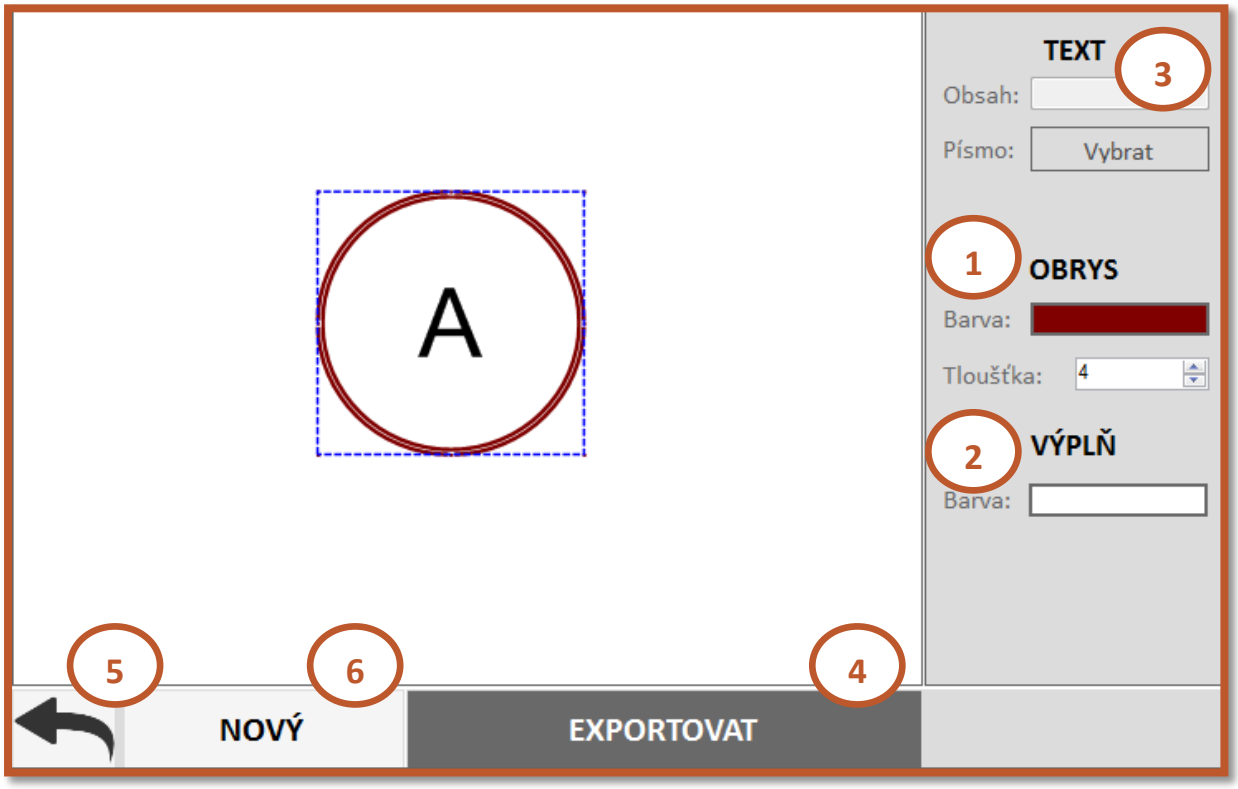
KROK 3 – ÚPRAVA VIZUALIZACE A EXPORT

Přijmutím rozpoznání diagramu dojde k jeho vizualizaci. Jednotlivé symboly se převedou na příslušné grafické objekty a zobrazí se na plátně aplikace. Po klepnutí do objektu dojde k jeho označení a můžete ho po plátně posunovat, měnit jeho velikost a nastavení vzhledu. Mezi nastavení vzhledu, která je možné upravit, patří následující:

a) **Obrys obrazce (1)** – jeho barva a tloušťka.
b) **Výplň obrazce (2)** – barva výplně (není možné nastavit výplň u šipek a textu).
c) V případě textu i **obsah (3)** a **písmo** textu.

Když jste s výsledkem spokojeni, můžete ho klepnutím na tlačítko **Exportovat (4)** uložit do souboru ve formátu PNG (obrázek). Opět bude zobrazen standardní dialog Uložit soubor.

Pro návrat k vytváření vstupního diagramu slouží tlačítko **Zpět (5)**, dojde ke ztrátě rozpoznání i vizualizace, budete moci upravit diagram a provést nové rozpoznání. Pokud chcete začít vytvářet úplně nový diagram, klepněte na tlačítko **Nový (6)** a pokračujte znovu od prvního kroku.



21. VIZUALIZACE DIAGRAMU.

PŘÍLOHA B – OBSAH CD

Přiložené CD osahuje následující složky a soubory týkající se této práce:

- **DiagramRecognition** – projekt se zdrojovými soubory aplikace.
- **DiagramRecognition.exe** – přeložená aplikace.
- **Dokumentace.html** – programátorská dokumentace k aplikaci.
- **Konečné automaty** – ukázkové soubory k rozpoznání
- **Vývojové diagramy** – ukázkové soubory k rozpoznání
- **Bakalářská práce.docx** – text práce ve formátu MS Word.
- **Bakalářská práce.pdf** – práce ve formátu PDF.