Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
študijný program: POČÍTAČOVÉ SYSTÉMY A SIETE

Podpora vzdelávania v predmete Špecifikačné a opisné jazyky

Dokumentácia k prototypu:
Tímový projekt I - zimný semester

Členovia tímu: Bc. Peter Koine, Bc. Šimon Hupka, Bc. Ján Turoň,
Bc. Michal Choleva, Bc. Emil Moťovský
Vedúca tímového projektu: Ing. Katarína Jelemenská, PhD.
Ročník, typ štúdia: 1., Inžinierske štúdium

1 Prototyp

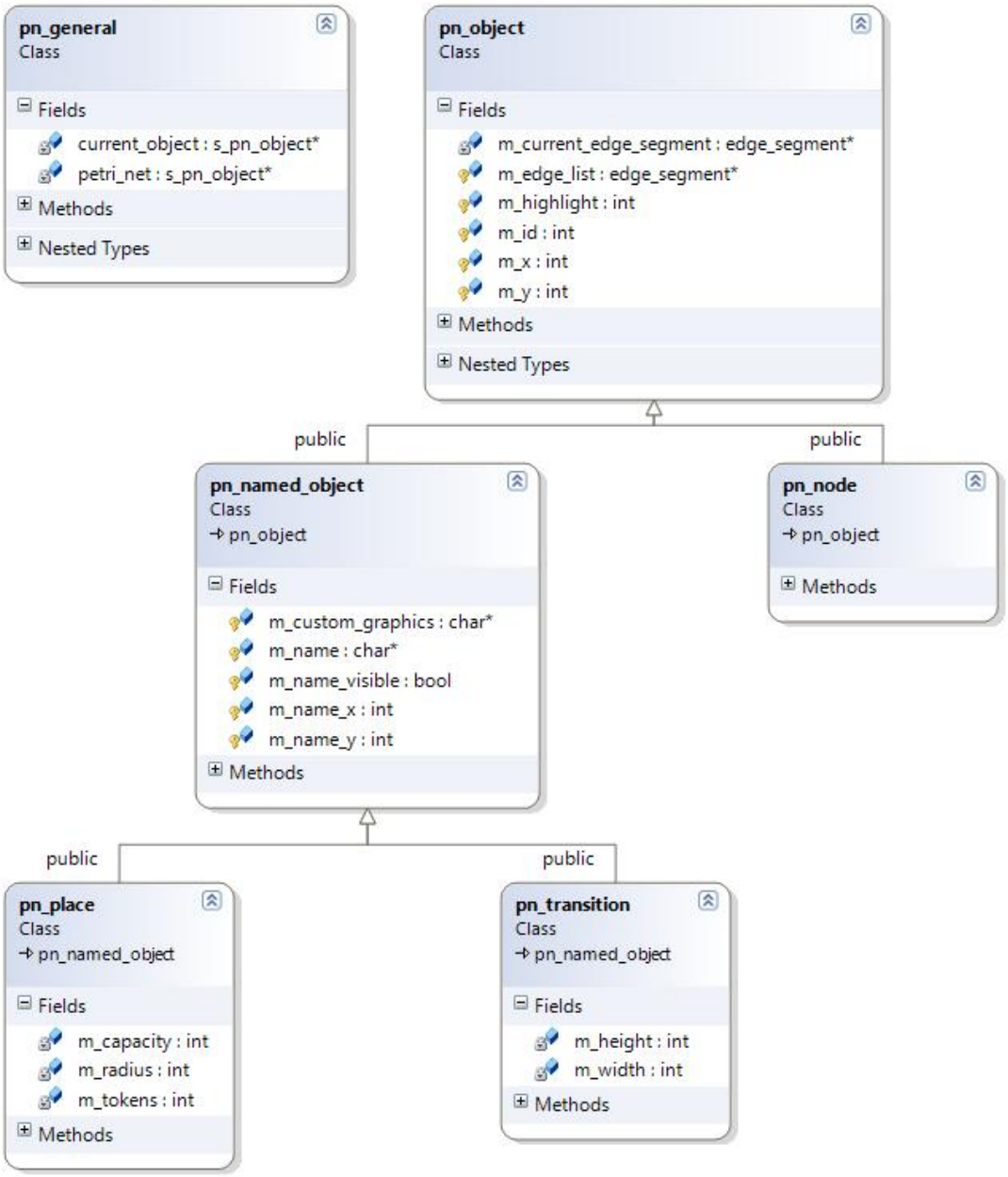
V prototype sa usilujeme overiť kľúčové a nové funkcie budúcej aplikácie. Zisťujeme nakoľko spoľahlivé je vytvárane Petriho siete kreslením na Tablet PC respektíve na bežnom PC s myšou. Zisťujeme ako verne dokážeme pomocou aplikácie nahradiť klasický papier. V ostatnom čase definujeme formát súborov. Konkrétne je potrebné:

- Implementovať robustný, objektovo orientovaný **interný model Petriho siete** poskytujúci funkcie podporujúce grafické znázornenie, analýzu vlastností Petriho siete, prípadne jej simuláciu.
- Implementovať finálne **používateľské rozhranie** alebo jeho časť. Minimálne umožňuje vytváranie Petriho siete pomocou interného modelu, jej zobrazenie a niektoré elementárne operácie.
- Implementovať **operácie pre vytváranie a úpravy** Petriho siete tak, aby z hľadiska používateľa v maximálnej miere korešpondovali s ekvivalentnými operáciami pri kreslení na papier.
- Implementovať **algoritmus pre rozoznávanie** kreslených objektov. Kresba používateľa sa prekladá na konkrétny objekt Petriho siete.
- Podrobne navrhnúť **formát súborov**, pričom treba zvážiť možnosť použiť totožný formát pre sieťovú komunikáciu. Súčasťou je čiastočná alebo úplná implementácia algoritmov pre súborové operácie do prototypu.

1.1 Fyzický model údajov

Pre potreby prototypu boli implementované iba tie entity, ktoré sa podieľajú na modelovaní Petriho siete. Tieto chápeme ako množinu objektov a sú reprezentované triedou „pn_general.“ Všeobecný objekt definovaný v logickom modeli zodpovedá triede „pn_object,“ od ktorej sme pomocou dedičnosti odvodili trojicu Miesto, Prechod, Uzol. Model Petriho siete neobsahuje žiaden objekt pre definovanie hrán. Tieto sú jednosmerné, chápané ako množina navzájom prepojených objektov. Prakticky sú riešené tak, že každý objekt obsahuje žiadnu, jednu alebo viac referencií na svojho nasledovníka. Začiatok a koniec hrany je miesto alebo prechod, ostatné objekty sú uzly.

Vlastný model povoľuje aj vytváranie nežiaducich konštrukcií ako napríklad prepojenie dvoch miest respektíve rozvetvenie hrany mimo prechodu. Toto správanie musí byť ošetrené samostatnou funkciou mimo modelu. Na druhej strane vďaka chýbajúcej implicitnej kontrole správnosti sú možnosti vytvárania Petriho siete rovnako neobmedzené ako pri kreslení na papier.



Obr. 21 - Triedy definované pre potreby prototypu, ich dedičné väzby a členské premenné.

1.1.1 Petriho sieť - pn_general

Zastrešuje všetky objekty prislúchajúce do jednej Petriho siete. Poskytuje funkcie pre manipuláciu s objektmi a navigáciu medzi nimi. Súčasťou je aj funkcia pre určenie objektu na základe súradníc. Objekty sú uložené v spájanom zozname, pričom jeho prvkom je štruktúra „s_pn_object“ obsahujúca jeden objekt („pn_object“) a referenciu na nasledujúci prvok spájaného zoznamu.

Trieda: pn_general		
Názov atribútu	*Typ*	*Opis*
petri_net	s_pn_object*	Počiatočný prvok spájaného zoznamu objektov.
current_object	s_pn_object*	Aktuálny prvok, s ktorým používateľ práve pracuje. Musí byť explicitne nastavený príslušnou funkciou.

1.1.2 Objekt - pn_object

Abstrakcia nad všetkými typmi objektov používanými v Petriho sieti. Definuje vlastnosti spoločné pre miesto, prechod a uzol, základné a doplnkové funkcie pre prístup k týmto premenným.

Pre každý objekt je definovaný zoznam hrán. Tento je riešený formou spájaného zoznamu zloženého z prvkov typu „edge_segment.“ Pre daný objekt sa ukladajú len hrany smerujúce od objektu. Hrana, zvyčajne zložená zo segmentov, nie je v zdrojovom objekte uložená celá ale vždy iba jeden z jej segmentov. Segment je pritom reprezentovaný referenciou na nasledujúci objekt hrany a váhou hrany (číslo typu „integer“).

Trieda: pn_object		
Názov atribútu	*Typ*	*Opis*
m_current_edge_segment	edge_segment*	Aktuálna hrana, s ktorou používateľ práve pracuje. Musí byť explicitne nastavená príslušnou funkciou.
m_edge_list	edge_segment*	Počiatočný prvok spájaného zoznamu hrán.

m_highlight	integer	Príznak zvýraznenia. Indikuje, že používateľ vybral objekt a chce s ním pracovať.
m_id	integer	Jednoznačný číselný identifikátor objektu.
m_x	integer	Horizontálna poloha objektu.
m_y	integer	Vertikálna poloha objektu.

1.1.3 Uzol - pn_node

Je jedným z troch reálne používaných objektov. Je odvodený od abstraktného objektu a tento nedopĺňa o žiadne ďalšie členské premenné alebo funkcie. Tak ako miesto a prechod poskytuje funkciu „kind“ na určenie typu objektu. Pre zjednodušenie implementácie a zrýchlenie behu programu môžeme uvažovať, že z každého uzla vychádza iba jedna hrana, čo v korektných sieťach platí. Pre zaistenie správnosti nášho predpokladu však potrebujeme dodatočne implementovať funkciu pre kontrolu správnosti Petriho siete.

1.1.4 Pomenovaný objekt - pn_named_object

Odvodená od triedy „pn_object,“ je abstrakciou nad miestom a prechodom. Tieto dva typy objektov majú niekoľko spoločných charakteristík, ktoré vyjadrujú pridané členské premenné. Tieto sú sprístupnené pomocou príslušných „set“ a „get“ funkcií.

Trieda: pn_named_object		
Názov atribútu	*Typ*	*Opis*
m_custom_graphics	char*	Názov súboru s používateľským obrázkom, ktorý sa vykreslí namiesto štandardného symbolu.
m_name	char*	Názov objektu.
m_name_visible	bool	Príznak zobrazenia menovky objektu.
m_name_x	integer	Relatívna poloha menovky vzhľadom k objektu – horizontálne.
m_name_y	integer	Relatívna poloha menovky vzhľadom k objektu – vertikálne.

1.1.5 Miesto - pn_place

Je druhým reálne používaným objektom. Postupne dedí vlastnosti objektu a pomenovaného objektu a pridáva vlastné členské premenné a funkcie. Obsahuje vlastnú implementáciu funkcie pre určenie súradníc takzvaného konektora. Týmto rozumieme bod, ktorého poloha sa voči tomuto objektu zostáva fixná, ale vzhľadom k ostatným objektom premenlivá. Prakticky sa táto funkcia používa pri umiestňovaní a otáčaní šípok, ktoré vždy smerujú do miesta.

Trieda: pn_place		
Názov atribútu	*Typ*	*Opis*
m_capacity	integer	Kapacita miesta. Nekonečno je určené hodnotou 2^15-1.
m_radius	integer	Polomer miesta.
m_tokens	integer	Počet značiek v mieste.

1.1.6 Prechod - pn_transition

Tretí reálny objekt je rovnako ako miesto odvodený od triedy „pn_named_object“ pričom pridáva špecifické členské premenné a funkcie pre prístup. Obdobne obsahuje vlastnú implementáciu funkcie pre určenie konektora.

Trieda: pn_transition		
Názov atribútu	*Typ*	*Opis*
m_height	integer	Výška prechodu.
m_width	integer	Šírka prechodu.

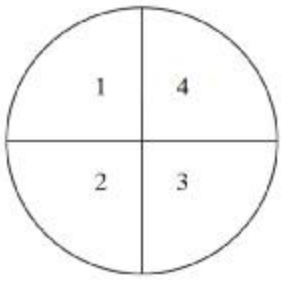
1.2 Významné algoritmy

Väčšina akcií sa vykonáva myšou priamo v kresliacej ploche čo viedlo z vzniku rozsiahlych funkcií pre obsluhu pohybu myši a stlačenia ľavého tlačidla, ako v stave stlačené tak uvoľnené. Obsahujú starostlivo navrhnuté rozhodovacie algoritmy, ktoré zabezpečujú rozlíšenie požiadavky používateľa. Na základe toho môže byť napríklad vytvorené nové miesto alebo presunutý niektorý z objektov. Momentálne je súčasťou týchto obslužných funkcií aj množstvo iných algoritmov, ktoré by sa mali nachádzať vo vlastných funkciách. Spomedzi všetkých implementovaných algoritmov si niektoré popíšeme.

1.2.1 Rozpoznávanie objektov

Zatiaľ je implementované iba rozpoznávania miesta. Ak sa týmto algoritmom nakreslený obrázok nerozpozná ako miesto, automaticky je vyhodnotený obrázok ako prechod. Tento systém však plánujeme samozrejme zmeniť. Okrem rozpoznávania miesta a prechodu chceme implementovať tiež rozpoznávanie iných gést ako napríklad krížik pre mazanie a podobne.

Samotný algoritmus na rozpoznávanie miesta vyplýva z teórie, ktorú sme si vymysleli samy. Táto teória uvažuje kružnicu ako základ, z ktorého vychádza. Kružnicu sme si rozdelili na zhodné kvadranty ako je zobrazené na Obrázku č.22.



Obr. 22 - Kvadranty kružnice

Ak si túto kružnicu predstavíme v dvojrozmernej súradnicovej sústave s osami x a y, a budeme pozorovať závislosti oboch súradníc v rámci všetkých kvadrantov, zistíme nasledujúce.

V prvom kvadrante smerom k druhému postupne klesajú obe súradnice. V druhom kvadrante smerom k tretiemu narastá súradnica x, ale súradnica y stále klesá. Ďalej v treťom kvadrante narastajú obe a nakoniec v štvrtom x klesá a y narastá.

Pri kreslení si do statického poľa ukladáme súradnice po ktorých sa pohybuje pero. Toto pole potom rozdelíme na štvrtiny a zisťujeme závislosti medzi x a y v jednotlivých skupinách. Keďže používateľ môže kružnicu nakresliť rôznymi smermi, musíme v každej štvrtine testovať všetky možnosti závislostí. Závislosť považujeme za nájdenú, ak zo všetkých bodov štvrtiny viac ako polovica spĺňa danú podmienku. Pokiaľ nájdeme aspoň tri závislosti zo štyroch, obrázok je rozpoznaný ako miesto.

1.2.2 Kontrola korektnosti siete

Chybné konštrukcie môžu vzniknúť prepájaním nesprávnych objektov, overenie sa teda zameriava výlučne na kontrolu hrán. Vykonáva sa bezprostredne pred vykonaním konkrétnej akcie – pridanie hrany. Prvým testom je overenie či nie sú prepojené dva objekty rovnakého typu (miesto s miestom alebo prechod s prechodom). Ďalej sa určuje či zdrojový alebo cieľový objekt hrany nie je uzol. Zabráni sa tým vetveniu alebo spájaniu hrán mimo prechodov a miest. Posledný test zisťuje či sa nevytvára duplicitná hrana porovnaním zdrojového a cieľového objektu. V prípade, že niektorý z týchto testov neprebehne úspešne, zamedzí sa pridaniu hrany. Pre zrýchlenie behu algoritmu sa uvažujú niektoré zjednodušenia, ktoré v korektnej sieti platia. V nekorektných sieťach vedú k vzniku nekonečných slučiek. V snahe zabrániť zlyhaniu celej aplikácie sme pridali jednoduché počítadlo, ktoré ukončí rizikové cykly ak prebehnú omnoho viac krát ako sa očakáva.

1.2.3 Vymazávanie súvisiacich objektov

Odstránenie objektu je elementárnou operáciou, definovanou v triede „pn_general.“ Napríklad pri odstránení miesta, je zbytočné ponechať hrany vedúce do a z tohto miesta. Tieto je teda potrebné vymazať. Odstránia sa všetky uzly v rámci tejto hrany a všetky referencie na odstránené objekty. Hrany vedúce z odstráneného objektu sa vymažú postupným odstránením všetkých uzlov v poradí ako nasledujú na hrane. Proces

končí nájdením objektu, ktorý je nasledovníkom odstraňovaného uzla a pritom nie je uzol. Tento sa neodstráni. Obdobne sa likvidujú hrany smerujúce do objektu. Rozdiel je v tom, že referencie sú jednosmerné a teda nepoznáme zdrojový objekt, miesto kde začíname s postupným odstraňovaním uzlov. Najprv teda musíme určiť, ktoré spomedzi všetkých hrán smerujú do odstraňovaného objektu.

1.3 Ostatné aspekty prototypu

Aplikácia je implementovaná ako dialógové okno s kresliacou plochou, niekoľkými ovládacími tlačidlami a stavovým riadkom. Väčšina akcií sa vykonáva pomocou myši a kliknutím jej ľavým tlačidlom, pri Tablet PC ťuknutím a pohybom pera priamo v kresliacej ploche. Eliminuje sa potreba tlačidiel a aplikácia vďaka tomu vernejšie nahrádza klasický papier.

Vytvorený prototyp umožňuje vytváranie Petriho siete kreslením, pričom o rozpoznanie objektu sa stará vyššie popísaný algoritmus alebo pomocou troch režimov myši (pridávanie miest, prechodov a hrán). Algoritmy spracúvajúce vstupy zatiaľ nie sú tolerantné, občas teda nesprávne interpretujú vstup a vykonajú odlišnú akciu ako požaduje používateľ. Zostavenú sieť možno ďalej modifikovať presúvaním jednotlivých objektov. Objekty môžeme vymazávať, pričom odstránené sú aj všetky súvisiace hrany. V špecifických prípadoch vymazanie objektu vedie k závažnému narušeniu korektnosti Petriho siete, avšak bez zlyhania aplikácie. Experimentálne sme implementovali funkciu pre overenie korektnosti siete. Testovali sme správanie sa aplikácie pri použití tejto funkcie ako aj bez nej. Pokiaľ je funkcia zapnutá, program až príliš napomáha k správnemu zostaveniu Petriho siete, preto pri písaní testov bude táto implicitne vypnutá. Na druhej strane existujú nekorektné modely, ktoré vedú k vytvoreniu nekonečných slučiek počas behu programu.

Pri návrhu formátu súborov a algoritmov pre ukladanie a načítavanie sme zistili, že u objektovo orientovaných programov existujú hotové, efektívne metódy pre súborové operácie. Rozpracované časti boli s komplikáciami dokončené a nakreslenú sieť môžeme uložiť do súboru a neskôr rekonštruovať načítaním z tohto súboru. Ukladá sa však iba zlomok z parametrov definovaných pre objekty.

Ďalšie podrobnosti o prototype, jeho používaní, spolu s vyobrazením používateľského rozhrania sa nachádzajú v prílohe „Používateľská príručka.“

1.4 Prototyp vo vzťahu k finálnej aplikácií

Vytvorením prototypu sme úspešne overili, že dokážeme verne nahradiť kreslenie Petriho siete na papier kreslením do aplikácie bežiacej na Tablet PC. Zároveň dokážeme poskytnúť alternatívny režim, vhodný pre bežné PC s myšou. Rozpoznávanie objektov a používateľských akcií zatiaľ nie je tolerantné. V budúcnosti bude potrebné vylepšiť algoritmus pre rozpoznávanie objektov a doplniť ho o nové symboly, napríklad krížik - preškrtnutie pre vymazanie objektu. Je potrebné doplniť toleranciu, tak aby aplikácia správne reprezentovala aj neúplné a nepresné ťahy.

Zostrojili sme spoľahlivú objektovú reprezentáciu Petriho siete, ktorú môžeme ďalej použiť vo finálnej aplikácii. Bude potrebné implementovať triedy „Úloha“ a „Test“ tak aby platilo, že úloha obsahuje Petriho sieť, prípadne ďalšie údaje a test je zložený z úloh.

Snaha o návrh vlastného formátu súborov sa predbežne ukázala ako zbytočná. Súborové operácie implementujeme pomocou predefinovanej triedy „CArchive,“ ktorá umožňuje jednoduché a spoľahlivé uloženie objektovo reprezentovaných údajov a následne ich spätné načítanie zo súboru. Formát ukladania údajov do súboru si pri tomto riešení určí trieda „CArchive“ a je pre nás bezpredmetné zaoberať sa týmto problémov.

Používateľské rozhranie prototypu je dočasné. Vo finálnej aplikácii bude vytvorené na základe predefinovaného SDI (Single Document Interface), ktoré lepšie zodpovedá súčasným trendom pri návrhu používateľského rozhrania. Myšlienka v maximálnej miere komunikovať s aplikáciou prostredníctvom myši zostáva zachovaná bez zmien.

Na základe doterajších skúseností s prototypom sme upravili a naplánovali ďalšie činnosti, vedúce k zostaveniu finálnej aplikácie.