

Modelovanie a riadenie systému automaticky
navádzaných vozidiel pre dopravu vo
výrobných procesoch
(Tímový projekt)

Štefan Krištofik, Filip Lörinc, Igor Sečanský,
Stanislav Panák

15. novembra 2007

Účel a rozsah dokumentu

Predkladaný dokument slúži ako dokumentácia ku:

- analýze problému
- špecifikácií požiadaviek riešenia
- hrubému návrhu riešenia

programového systému na modelovanie a riadenie systému AGV pre dopravu vo výrobných procesoch vytváraného v rámci predmetu Tímový projekt I.

Použité skratky

AGV – Automated Guided Vehicles

FMS – Flexible manufacturing system

CNC – Computer controlled machines

PS – Petriho siete

Obsah

| | | |
|----------|---|-----------|
| 1 | Analýza problému | 5 |
| 1.1 | Úvod | 5 |
| 1.2 | Flexibilné výrobné systémy | 8 |
| 1.3 | Automaticky navádzané vozidlá | 9 |
| 1.3.1 | Zónová kontrola | 11 |
| 1.3.2 | Predvídavá kontrola | 11 |
| 1.3.3 | Kombinovaná kontrola | 11 |
| 1.4 | Riadenie systému | 11 |
| 1.5 | Systémy s AGV | 12 |
| 1.6 | Plánovanie AGV | 12 |
| 1.7 | Smerovanie AGV | 13 |
| 1.8 | Nežiadúce okolnosti | 13 |
| 1.9 | Smerovacie algoritmy | 15 |
| 1.10 | Smerovacie algoritmy pre topológie so všeobecnými cestami . . | 15 |
| 1.10.1 | Statické smerovacie algoritmy | 15 |
| 1.11 | Petriho siete | 18 |
| 1.11.1 | Analýza Petriho sietí | 20 |
| 1.11.2 | PNDesigner | 20 |
| 1.11.3 | CitectHMI | 20 |
| 1.12 | Grafy udalostí | 23 |
| 2 | Špecifikácia požiadaviek | 25 |
| 2.1 | Špecifikácia funkcií systému | 26 |
| 3 | Hrubý návrh riešenia | 29 |
| 3.1 | Cyklus programu | 29 |
| 3.2 | Vizualizácia | 29 |

| | |
|------------------------------|-----------|
| <i>OBSAH</i> | 3 |
| A Metóda Free Windows | 34 |
| B Preberací protokol | 45 |

Zoznam obrázkov

| | | |
|------|--|----|
| 1.1 | AGV | 5 |
| 1.2 | Špičkové laboratórne vozíky | 6 |
| 1.3 | Vozíky v tlačiarenskom priemysle | 6 |
| 1.4 | Vysokozdvížné vozíky vo výrobných halách | 7 |
| 1.5 | Model autíčok a robotického ramena | 8 |
| 1.6 | Fenomény pri plánovaní a smerovaní AGV | 14 |
| 1.7 | Porovnanie algoritmov | 17 |
| 1.8 | Petri Net Designer | 21 |
| 1.9 | Časť vývojového prostredia Citect | 22 |
| 1.10 | EGRET – Graf udalostí | 23 |
| 2.1 | Model prípadov použitia | 27 |
| 3.1 | Vizualizačný framework | 30 |
| 3.2 | Diagram tried vizualizačného frameworku | 31 |
| 3.3 | Životný cyklus programu | 32 |

Kapitola 1

Analýza problému

1.1 Úvod

Modelovanie výrobných procesov, pracovných postupov, komunikácie v sie-
ti, dopravnej situácie a mnoho ďalších vecí sa dá realizovať spoločnými
prostriedkami, ktoré zvyčajne vyúsťujú do reprezentácie pomocou rôznych



Obrázok 1.1: AGV

orientovaných grafov, ktoré sa navy-
še môžu dynamicky meniť, či už v
závislosti od času, alebo od výskytu
rozličných udalostí. Na vizualizáciu
sú vhodné napríklad grafy udalostí, s
ktorými sme sa mali možnosť stret-
núť na predmete *modelovanie a si-
mulácia* v podobe nástroja *EGRET*,
alebo Petriho siete, ktoré boli spo-
menuté v záverečných prednáškach
zo *špecifikačných a opisných jazykov*.

Petriho siete, okrem toho že prob-
lém vizualizujú, slúžia napríklad aj

na dokazovanie rôznych zacyklení, uviaznutí, alebo na dokázanie korektnosti
fungovania systému. Táto problematika je však veľmi náročná. Uplatnenie
tu nachádzajú aj grafové algoritmy. Napríklad na obrázku 1.1 vidno špičkový
automaticky navádzaný laboratórny vozík. Vozíky sú použiteľné v laborató-
riách alebo v nemocniciach na rôznych oddeleniach (biochémia, mikrobioló-
gia, imunológia, hematológia, farmácia, rádiológia, virológia, histológia, ...),

bezpečne prechádzajú popri personále, dokážu manévrovať v úzkych priestoroch, samé dokážu otvárať a zatvárať dvere (obrázok 1.2), pohybujú sa



Obrázok 1.2: Špičkové laboratórne vozíky

po poschodiach využívajúc existujúce výťahy, sú v prevádzke 24 hodín denne, 7 dní v týždni a majú samodobíjanie batérií. Ľahko sa programujú a jednoducho nasadzujú.



Obrázok 1.3: Vozíky v tlačiarenskom priemysle

Ďalšími aplikáciami AGV (Automated Guided Vehicles) sú napríklad vysokozdvížné vozíky používané v tlačiarenskom priemysle na prepravu papierových kotúčov (obrázok 1.3) zo skladových priestorov ku tlačiarenskému stroju (vozík papierový kotúč dopraví, aj nasadí), alebo automatické vysokozdvížné vozíky požívané vo výrobných halách prípadne skladoch na prekladanie paliet, zabaleného tovaru a pod. (obrázok 1.4). Vozíky sa dokážu

orientovať v priestore (použitie senzorov, resp. kamerového systému s rozpoznávaním), vyhýbajú sa prekážkam, sú schopné identifikovať prepravovanú vec, prípadne vzájomne komunikujú (riešenie uviaznutí a zacyklení). Už zo-



Obrázok 1.4: Vysokozdvížne vozíky vo výrobných halách

strojenie a naprogramovanie reálneho modelu (obrázok 1.5) je veľmi náročné, preto sa obmedzíme na jednoduchý počítačový model, v ktorom budú vozidlá, resp. prepravované objekty reprezentované jednoduchými geometrickými objektami (krúžky, guľôčky, prípadne ich kompozícia propomínajúca autíčko a pod.). Trajektórie dráh, po ktorých budú vozidlá chodiť, budú reprezentované pomocou orientovaného grafu, ktorého vrcholy budú zastávky pre vozidlá (prekladiská, parkovacie miesta, čerpacie stanice a pod.) Program bude zobrazovať aktuálnu polohu vozidiel v čase a bude predstavovať akýsi velín, prípadne bude obdobou navigátorského systému. Ďalšou jeho funkciou bude optimalizácia. Bude sa jednať o nachádzanie najkratších ciest a podobne.



Obrázok 1.5: Model autíček a robotického ramena

Na nižšie uvedených linkách sa nachádzajú krátke videá, na ktorých možno vidieť fungovanie automaticky navádzaných vozidiel, o ktorých sme hovorili:

Laboratórne vozíky :

<http://www.youtube.com/watch?v=mgp1cvkDPTI>

Vozíky v tlačiarenskom priemysle :

<http://www.youtube.com/watch?v=iywjAVshnWg>

Vysokozdvížné vozíky vo výrobnjej hale :

<http://www.youtube.com/watch?v=hLwI-pHUvRQ>

Model s robotickým ramenom :

<http://www.youtube.com/watch?v=KnJljQ08LHo>

1.2 Flexibilné výrobné systémy

Flexibilný výrobný systém – *Flexible Manufacturing System* (FMS) – ide vlastne o viac intuitívny pojem ako o niečo hmatateľné. FMS je v podstate myšlienka, že rýchle je lepšie a používa stroje na výrobu produktov. Namiesťto využívania ľudských síl na vykonávanie opakujúcich sa úloh je použitý stroj, ktorý danú úlohu vykonáva 24 hodín denne. FMS využíva tzv. *computer numerical controlled machines* (CNC), čo sú počítačom riadené stroje, na vytvorenie tzv. *pracovných buniek*. Každá takáto bunka potom vykonáva špecifickú úlohu, čím napomáha svojim dielom k vyrobeniu produktu.

FMS sú rýchle a efektívne, na druhej strane však nie sú veľmi lacné a je potrebné mať veľa drahých strojov pre začatie výroby. Z tohto dôvodu veľa spoločností volí radšej cestu čiastočnej náhrady výroby systémom FMS. V takomto prípade hovoríme o vytváraní tzv. flexibilných výrobných buniek. Táto bunka(y) vyrobí časť finálneho produktu, zatiaľ čo ostatné časti sú vyrobené inými metódami. Často nastáva prípad, že niekoľko vozidiel typu AGV (vysvetlené nižšie) je vyhradených za účelom prepojenia jednotlivých výrobných buniek [4].

1.3 Automaticky navádzané vozidlá

Automatic Guided Vehicle (AGV) - automaticky navádzané vozidlo je mobilné robotické vozidlo ovládané počítačom používané v priemysle, ktoré slúži na premiestňovanie rôznych predmetov z jedného miesta na iné vo výrobných halách alebo v skladoch a pod. Takéto vozidlá sú vybavené zvyčajne elektromagnetickými alebo optickými zariadeniami, ktoré umožňujú ich automatické navádzanie. Takto vybavené vozidlá sú potom schopné pohybovať sa po predurčených trasách, zastavovať na určených miestach, a plniť ďalšie rozličné funkcie, ktoré sú na ne kladené počas prevádzky [4].

AGV pomáhajú znižovať náklady na výrobu a zvyšovať efektívnosť výrobného systému. Sú napríklad schopné ťahať predmety za sebou v malých vlečkách, na ktoré sa dokážu samé autonómne zahákovávať. Na týchto vlečkách je možné prenášať surový materiál do linky, kde je už pripravený na výrobu. AGV dokážu taktiež skladovať objekty na vrstvách alebo ich umiestňovať na poličky. Niektoré AGV dokonca používajú technológiu ako vysokozdvížné vozíky, kde sa pod predmety podsunú kovové vidlice, ktoré sa potom zdvíhajú do príslušnej výšky. Tento prístup nájde uplatnenie najmä pri práci v sklade. AGV môžu slúžiť aj na prepravu medicínskeho materiálu v urgentných prípadoch. Ich použitie je široké [3].

AGV sa stávajú stále populárnejšími v automaticky riadených systémoch, vo flexibilných výrobných systémoch a dokonca v systémoch s prepravou kontajnerov, napríklad v prístavoch. V predošlých desaťročiach sa technológii AGV venovalo množstvo výskumov a štúdií a bol zaznamenaný významný progres. Napríklad technológii plánovania (*scheduling*) a smerovania (*routing*) AGV sa v poslednom období venovalo viacero autorov. Navrhnutých bolo niekoľko algoritmov riešiacich tieto problémy, avšak väčšina existujúcich riešení je vhodná len pre systémy s malým počtom AGV. V súčasných

systémoch, kde počet vozidiel AGV sa môže pohybovať až okolo sto a viac, je nutné zaviesť efektívne algoritmy plánovania a smerovania vozidiel na riešenie zvýšeného počtu prípadov, kedy dochádza k súpereniu pri využívaní zdrojov medzi AGV (napríklad konflikty typu keď viacero vozidiel chce využiť tú istú cestu).

Takisto je potrebné zvyšovať celkový výkon takýchto systémov, čo môže byť dosiahnuté len inovatívnymi myšlienkami, postupmi a stratégiami ako napríklad paralelné spracovanie a podobné inovatívne prístupy. Na tomto poli sú teda v súčasnosti možnosti otvorené pre budúce výskumy a štúdie [3].

AGV v systémoch FMS sú používané na premiestnenie objektu z bodu A do bodu B. AGV sa orientuje vo výrobných priestoroch použitím senzorov. Použiť sa môžu dva typy senzorov: drôtové a bezdrôtové (napríklad navádzanie laserom, gyroskopické navádzanie, ...) [4]. Rozoznávame viaceré druhy AGV:

- ťažné – najstarší typ, stále však používaný
- nakladacie – majú plošiny na nakladanie materiálu, môžu byť zdvíhané, znižované, sú rôzne typy
- paletizované – na prepravu materiálov na paletách
- vidlicové – zdvíhanie materiálov, ukladanie materiálov do poličiek
- nízkonákladové – na transport menších, ľahších častí, používajú sa v prostrediach s obmedzenými priestorovými možnosťami
- výrobné – nízkonákladové, používajú sa pri procesoch sériovej výroby pri linkách [4]

Ak FMS má viacero AGV, je nutné nejakým spôsobom zabezpečiť kontrolu dopravného toku, aby napríklad AGV navzájom do seba nenarážali a nevznikli podobné problémy. Na riešenie kontroly dopravy [4] existujú nasledovné metódy:

- zónová kontrola
- predvídavá kontrola
- kombinovaná kontrola

1.3.1 Zónová kontrola

Táto metóda kontroly dopravy v FMS je obľúbená vo väčšine prostredí nakoľko sa ľahko inštaluje a rozširuje. Zónová kontrola používa bezdrôtový vysielateľ na prenos signálu v nemennom priestore. Každé AGV má zariadenie na príjem signálu a jeho spätné vysielanie k vysielateľu. Ak oblasť je voľná, AGV obdržia signál *clear* (voľný) a je im umožnený vstup do oblasti. Ak v oblasti je nejaké AGV, potom ostatné AGV, ktoré sa snažia do tejto oblasti vstúpiť, obdržia signál *stop* (zastaviť) a musia počkať. Po východe AGV z oblasti obdrží jedno z čakajúcich AGV signál *clear*. Inou možnosťou ako zaviesť zónovú kontrolu je vybaviť každé AGV svojim vlastným vysielateľom a prijímačom. Potom takto vybavené vozidlo je schopné po vstupe do oblasti rozoslať ostatným vozidlám v blízkom okolí signál typu *do not enter* (nevstupovať) [4].

1.3.2 Predvídavá kontrola

Tento spôsob používa špeciálne senzory na predchádzanie kolíziám s inými AGV v oblasti. (zvukové, optické senzory, senzory fyzického kontaktu). Senzory fyzického kontaktu sú namontované na AGV ako záloha. Takéto riešenie je však náročné čo sa týka inštalácie aj práce s ním [4].

1.3.3 Kombinovaná kontrola

Ide o prípad, kedy je použitá kombinácia oboch vyššie spomenutých metód. AGV majú senzory predchádzania kolíziám aj senzory zónovej kontroly. Kombináciou týchto sa predíde kolíziám v každom prípade. Pri bežnej prevádzke sú použité senzory zónovej kontroly s využitím senzorov predchádzania kolízie ako poistka. Napríklad v prípade výpadku zónovej kontroly potom predvídavá kontrola predíde prípadným kolíziám [4].

1.4 Riadenie systému

Výroba s využitím AGV musí mať určitý stupeň kontroly nad týmito vozidlami. Toto je možné zabezpečiť troma metódami:

- panel lokátora – jednoduchý panel, kde vidno, kde sa AGV nachádza

- farebný displej CRT – v reálnom čase zobrazuje pozície všetkých AGV a ich stav (napätie batérie, ID, a pod.)
- centralizované logovanie – použije sa na uchovávanie histórie o pohybe všetkých AGV v systéme (kde boli, kadiaľ išli a podobne). Takto nazbierané údaje sa uchovávajú v nejakom centrálnom bode a je možné ich napríklad vytlačiť za účelom nejakej kontroly príp. inými účelmi [4].

1.5 Systémy s AGV

Automated Guided Vehicle System (AGVS) – systémy s automaticky navádzanými vozidlami sú také priemyselné systémy, v ktorých automaticky navádzané vozidlá zohrávajú dôležitú úlohu ako hlavné prostriedky (nástroje) na transport materiálov. AGVS môžeme definovať aj ako počítačom riadený systém, ktorý obsahuje navzájom nezávisle adresovateľné vozidlá bez vodiča (to sú AGV) pohybujúce sa po dopredu zadanovej sieti transportných ciest [3]. Presne tak, ako počítačový systém, aj systémy s automaticky navádzanými vozidlami sú typicky zložené z dvoch hlavných kooperujúcich podsystémov - hardvér a softvér. Hardvér zahŕňa fyzické komponenty ako samotné vozidlá, cesty, senzory, navádzacie systémy, ovládacie prvky a pod. Softvér zahŕňa rôzne prístupy a algoritmy na systematické manažovanie hardvérových prostriedkov celého AGVS. Tieto prístupy a algoritmy napomáhajú harmonickému fungovaniu AGVS a k ich vysokej efektívnosti. V posledných desaťročiach však je pokrok v oblasti hardvérového vybavenia AGVS oveľa výraznejší ako pokrok v oblasti vývoja softvéru. Toto zaostávanie softvérového vybavenia začalo byť problémom až v posledných rokoch pri zvyšovaní počtu AGV v systémoch AGVS. Taktiež problémy plánovania a smerovania AGV, ktoré boli v minulosti považované za triviálne, sa stali dôležitými len v poslednom období [3].

1.6 Plánovanie AGV

Pod plánovaním (*scheduling*) AGV môžeme chápať činnosť, keď pošleme skupinu AGV vykonať sériu jednoduchých úloh typu napríklad zdvihnutie/položenie predmetu za cieľom dosiahnuť nejaký konkrétny cieľ pri daných obmedzeniach [3]. Príklady cieľov plánovania:

- ukončiť úlohu do určitej doby (deadline)
- ukončiť úlohu za minimálny možný čas
- minimálna prejdená vzdialenosť všetkých AGV
- minimálna doba nečinnosti (idle time) všetkých AGV
- plnenie úlohy (napr. produkcia výrobkov) s čo najnižším možným počtom AGV

Plánovanie zahŕňa aj nasledovné problémy:

- vehicle selection problem (vehicle dispatching problem) – z množiny momentálne voľných vozidiel vyberáme jedno, ktorému úlohu pridáme
- task selection problem – z množiny úloh vyberieme jednu, ktorú budeme niektorým vozidlom riešiť

1.7 Smerovanie AGV

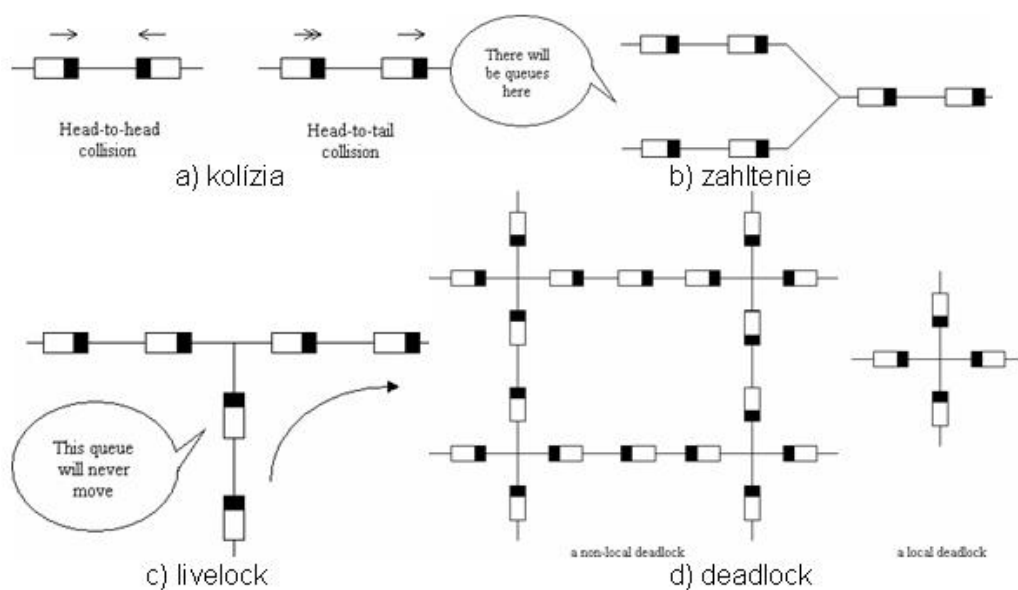
Pod smerovaním (*routing*) AGV rozumieme hľadanie optimálnej a uskutočniteľnej cesty pre každé AGV. Tento proces [3] zahŕňa tri aspekty:

1. Musí sa zistiť, či vôbec nejaká cesta medzi zdrojom a cieľom cesty existuje.
2. Cesta musí byť uskutočniteľná, to znamená že nesmie dôjsť ku upchaniu cesty ani ku konfliktom medzi vozidlami.
3. Cesta musí byť optimálna, alebo aspoň čiastočne optimálna (napr. minimalizovať čas nečinnosti vozidla).

1.8 Nežiadúce okolnosti

Napriek tomu, že problémom plánovania a smerovania AGV bolo venované množstvo výskumného úsilia, stále chýbajú prijateľné všeobecné riešenia a to kvôli rôznorodosti aplikácií. Navrhnuté postupy a algoritmy sú zvyčajne vhodné len pre špecifické prostredia, alebo sa venujú riešeniu problémov, ktoré sa objavujú len v osobitých prípadoch. Pri plánovaní a smerovaní AGV sa bežne stretávame s nasledovnými problémami (tzv. *fenoménni*) [3]:

- *kolízie* – ak sa viacero AGV pokúša využiť ten istý segment cesty v tom istom čase. Príklad kolízie je na obr. 1.6 a).
- *zahľtenie* – nastáva tam, kde počet príchodších vozidiel je väčší ako počet požiadaviek. Miera zahľtenia musí byť znížená alebo musí byť odstránené úplne, nakoľko znižuje celkovú efektívnosť systému a môže viesť k ďalším problémom. Príklad zahľtenia je na obr. 1.6 b).
- *livelock* – špeciálny prípad, ak priorita premávky na jednej ceste na križovatke je neustále vyššia ako priorita premávky na inej ceste. Príklad takejto situácie je znázornený na obr. 1.6 c).
- *deadlock* – tento prípad nastane, ak viacero AGV čaká na uvoľnenie zdroja (ktoré nikdy nenastane), ktorý je okupovaný inými AGV. Dva príklady takejto situácie sú na obr. 1.6 d).



Obrázok 1.6: Fenomény pri plánovaní a smerovaní AGV

1.9 Smerovacie algoritmy

Existujúce práce v tejto oblasti môžeme rozdeliť do troch kategórií [3]:

- smerovacie algoritmy pre topológie so všeobecnými cestami
- optimalizácia ciest
- smerovacie algoritmy pre topológie so špecifickými cestami

Práce v prvej kategórii zvyčajne transformujú problém smerovania na teoretický problém z oblasti grafov a používajú prístupy ako *Dijkstrov algoritmus hľadania najkratšej cesty* na nájdenie optimálnych ciest pre AGV. Takéto algoritmy sú obyčajne veľmi náročné na výpočtový čas v prípade smerovania viacerých vozidiel. Práce v druhej kategórii sa zaoberajú použitím na mieru šitých optimalizačných techník ako napr. integerové programovanie na optimalizovanie sietí ciest namiesto hľadania efektívnejších smerovacích algoritmov. Pretože však v týchto prípadoch je výpočtová náročnosť veľmi vysoká, veľkosť optimalizovanej siete je obyčajne malá a umožňuje obsiahnuť zhruba do 20 AGV. Čo sa týka prác v tretej kategórii, tu sieť tratí má špecifickú topológiu, v ktorej sa nachádzajú napríklad viacnásobné slučky, preto na smerovanie AGV sú potrebné vysoko špecializované algoritmy [3].

Bližšie sa pozrieme len na prvú kategóriu smerovacích algoritmov, pretože pre triedu zónovo riadených vozidiel budeme používať jeden z algoritmov takéhoto typu.

1.10 Smerovacie algoritmy pre topológie so všeobecnými cestami

Algoritmy v tejto kategórii sa zameriavajú na nájdenie uskutočniteľných ciest pre AGV pričom do úvahy sa neberie topológia siete dráh. Snažia sa nájsť univerzálne riešenie smerovacieho problému. Základnou úvahou je poskytnúť bezkonfliktné časovo najkratšie možné riešenia smerovania pre vozidlá AGV [3].

1.10.1 Statické smerovacie algoritmy

Prvé výsledky (*Broadbent* 1985) v tejto oblasti využívali Dijkstrov algoritmus na nájdenie najkratšej cesty v grafe na vygenerovanie matice, ktorá popisuje

vala okupáciu ciest vozidlami v čase. Potenciálne konflikty medzi vozidlami, typu *head-to-head*, *head-to-tail* a kolízie na križovatkách boli detegované porovnaním časov okupácie dráh. Head-to-head konflikty boli vyriešené nájdením inej najkratšej cesty pričom konfliktný segment sa vynechal. Konflikty na križovatke a head-to-tail boli vyriešené spomalením alebo pozastavením vozidla, aby mohlo skôr naplánované vozidlo prejsť inkriminovaný úsek ako prvé. Dalším výrazným posunom bolo nahradiť doteraz používané jedno-smerné cesty pri smerovaní AGV obojsmernými. Výhody takéhoto riešenia čo sa týka využitia vozidiel a produktivity celého systému sú zrejmé. Egbeľu a Tanchoco v roku 1986 boli schopní takýmto spôsobom zmenšiť počet potrebných AGV v systéme pričom sa zároveň zvýšila produktivita. O takýchto systémoch ale platí, že môžu byť veľmi komplexné, pretože môže dôjsť k sporom o určité obojsmerné úseky. Smerovacie algoritmy musia byť schopné eliminovať potenciálne konflikty, kolízie aj deadlocky. Spomínaní Egbeľu a Tanchoco ako prví v r. 1986 navrhli použitie obojsmerných ciest pri smerovaní AGV. V ich práci však nebol navrhnutý žiaden algoritmus na optimalizáciu smerovania po obojsmerných cestách. Prvým, kto predložil algoritmus na smerovanie AGV po obojsmerných cestách bol Daniels v r. 1988. Bol to PSP algoritmus (*partitioning shortest-path algorithm*). Realizovateľnosť a správnosť algoritmu bola v tejto práci matematicky dokázaná. Algoritmus bol schopný nájsť bezkonfliktnú časovo najkratšiu cestu pre novo pridané AGV bez toho, aby sa cesty ostatných vozidiel menili. Ak však bol niektorý segment rezervovaný niektorým vozidlom, potom novo prichádzajúce vozidlo tento úsek nemohlo vôbec využiť hoci tento úsek bol nedostupný len po určitú dobu (čiže kým to vozidlo tadiaľ prechádzalo). Spomínané problémy s Danielsovým algoritmom sa snažil vyriešiť Huang a v r. 1989 navrhol značkovací algoritmus pre smerovanie jediného AGV v systéme s obojsmernými cestami. Algoritmus skonvertoval každú hranu (fyzickú) z originálnej siete na uzol s dvoma hranami spájajúcimi originálne 2 uzly v neskonvertovanej sieti. Takto je získaná skonvertovaná sieť a pridelia sa značky voľným časovým úsekom (oknám) definovaným pre každý uzol. Porovnávaním týchto značiek každého uzla bolo možné získať časovo najkratšiu cestu, ak taká existovala. Dalším vylepšením metód navrhnutých Danielsom a Huangom bola práca Kima a Tanchoca, ktorí v r. 1991 prezentovali algoritmus nájdenia bezkonfliktnej časovo najkratšej cesty pre AGV v sieti s obojsmernými cestami. Ich algoritmus bol založený na Dijkstrovom algoritme hľadania najkratšej cesty. Pre každý uzol sa uchovával zoznam časových okien rezervovaných naplánovanými vozidlami a zoznam voľných časových okien, ktoré

boli k dispozícii pre plánovanie ďalších vozidiel. Predstavený tu bol koncept grafu časových okien, v ktorom množina uzlov reprezentovala voľné časové okná a množina hrán reprezentovala dosiahnuteľnosť medzi voľnými časovými oknami. Potom algoritmus vlastne smeroval vozidlá cez jednotlivé voľné časové okná v grafe časových okien namiesto skutočných uzlov v sieti (podrobnejšie v metóde *Free windows*) Kim a Tanchoco svoju metódu vylepšili v r. 1993 o princíp tzv. konzervatívnej krátkozrakkej stratégie, kde je uvažované naraz len jedno vozidlo v čase a všetky predošlé naplánovania sú striktné rešpektované a následné cestovné plány sú vozidlám prideľované len až keď vozidlo sa stane nečinným (*idle*) [3]. Na obrázku 1.7 je tabuľka s porovnaním vyššie spomínaných algoritmov.

| | Broadbent 1985 | Daniels 1988 | Huang 1989 | Kim a Tanchoco 1991, 1993 |
|---------------------|---|--|--|---|
| Riešené problémy | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV |
| Základný algoritmus | Dijkstrov algoritmus hľadania najkratšej cesty | Deliaci algoritmus hľadania najkratšej cesty (PSP) | – | Dijkstrov algoritmus hľadania najkratšej cesty, konzervatívna krátkozraká stratégia |
| Zložitosť výpočtu | $O(N^2)$ (priemerný prípad) | $O(N \times A)$ (priemerný prípad) | $O((N+A)^2 \log(N+A))$ (priemerný prípad) | $O(\sqrt{A}N^2)$ (najhorší prípad) |
| Smer ciest | obojsmerné | obojsmerné | obojsmerné | obojsmerné |
| Výhody | Lahko realizovateľné | Lahko realizovateľné a rýchlejšie ako Broadbentovo | Časové okná sú použité pre každý uzol, využitie úsekov ciest je zvýšené | Lahko realizovateľné a kontrolovateľné, rýchle |
| Nevýhody | Zložitosť výpočtu, slabé využitie úsekov ciest | Zložitosť výpočtu, slabé využitie úsekov ciest, môže spôsobiť zlyhanie pri hľadaní ciest, ktoré existujú | Zložitosť výpočtu, veľké množstvo dát o konvertovanej sieti na udržiavanie | Zložitosť výpočtu, veľké množstvo dát o sieti na udržiavanie |

Legenda:

N – počet uzlov v sieti ciest A – počet hrán v sieti ciest v – počet vozidiel

Obrázok 1.7: Porovnanie algoritmov

Pre riešenie nášho projektu sme vybrali metódu *Free Windows* (pozri dodatok) popísanú *Kimom* a *Tanchocom* z r.1991 [1], ktorá je založená na Dijkstrovom algoritme hľadania najkratšej cesty v orientovanom grafe a využíva princíp časových okien na smerovanie AGV pohybujúcich sa po obojsmerných cestách v sieti. Pri analýze danej problematiky sme sa zaoberali *Petriho sieťami* a *udalostnými grafmi*.

1.11 Petriho siete

Petriho sieť je grafickým a matematicky opísateľným nástrojom, vhodným pre modelovanie a analýzu systémov diskrétnych udalostí. Ako grafický nástroj sú využívané ako pomôcka pre vizuálnu komunikáciu, podobne ako blokové diagramy, diagramy tokov a sietí. Navyše pomocou značiek dokážeme pomerne ľahko simulovať dynamické a súbežné udalosti v systéme. Sú tiež efektívny matematický nástroj, keďže ich dokážeme popísať pomocou stavových rovníc, algebraických rovníc, či inými matematickými modelmi, ktoré pokrývajú správanie sa systémov. Skrátene ich môžeme použiť pre všetky systémy, ktoré môžeme charakterizovať ako paralelné, distribuované, asynchrónne, nedeterministické, stochastické, súbežné. Majú nespočetné aplikácie v oblasti spracovávaní dát, distribuovaných systémov, vyhodnocovania výkonu alebo riadenia zložitých procesov.

Petriho sieť pozostáva z miest, prechodov a hrán, ktoré ich ako orientované úsečky spájajú. Vstupné hrany spájajú jednotlivé miesta s prechodmi, zatiaľ čo výstupné hrany začínajú v prechodoch a končia v miestach. Podobne môžeme rozdeliť aj jednotlivé miesta vzhľadom na konkrétny prechod na vstupné (s prechodom ho spája vstupná hrana) a výstupné (s prechodom ho spája výstupná hrana). Miesta môžu obsahovať značky, pomocou ktorých vzniká značkovanie ktorým môžeme popísať konkrétny stav modelovaného systému. Prechody sú dynamické entity pomocou ktorých dokážeme modelovať akcie aké môžu nastať v reálnom systéme a tak meniť stav v ktorom sa Petriho sieť (modelovaný systém) nachádza. Prechody sa spúšťajú len za špecifických podmienok, keď sú splnené všetky podmienky pre ich aktiváciu. Prechod je aktivovaný vtedy, ak sa vo všetkých vstupných miestach nachádza aspoň minimálny počet značiek, ktorý je určený príslušnou vstupnou hranou (jej kardinalitou).

Keď je daný prechod spustený, daný počet značiek vo vstupných miestach je odobraný a následne sa pridávajú značky do všetkých výstupných miest.

Ich množstvo zase závisí od kardinality výstupných hrán. Základná hodnota hrán je 1 a zvyčajne sa číselne nezobrazuje. Pomocou spúšťania prechodov v určitom poradí za určitého počiatočného značkovania môžeme skúmať danú Petriho sieť a určovať konkrétne vlastnosti.

Všeobecné Petriho siete neumožňujú modelovať to, že nejaká operácia trvá určitý čas, či hierarchiu systému. Takisto, počas simulácie reálnych systémov, na označenie miest vo viacerých prípadoch prirodzené čísla nepostačujú. Práve z tohoto dôvodu existuje viacero modifikácií, využívajúcich obohacovanie všeobecných vlastností o nové za účelom efektívnejšieho modelovania. Medzi tie známejšie patria:

- Deterministické časované PS – umožňujú popísať systém závislý od času, pričom berú do úvahy skutočnosť, že medzi začiatkom a koncom operácie prejde určitý čas.
- Stochastické PS – definujú stochastické časové oneskorenia prechodov.
- Farebné PS – siete tohto typu obsahujú značky rôzneho typu, ktoré sú navzájom rozlíšiteľné (graficky pomocou farby). Jednotlivým prechodom je priradená množina farieb, vzhľadom na ktoré môže byť aktivovaný. Hrany zase obsahujú funkciou, ako sa mení farba značky počas manipulácie s ňou. Značkami tohoto typu môžeme modelovať rozličné dátové typy a podobne.
- Spojité PS – sú realizované aplikáciou reálnych čísiel na označovanie miest.
- Hybridné PS a ďalšie ...

Značnou výhodou Petriho sietí je ich priama podpora pre analýzu rozličných vlastností modelovaného systému. Tieto vlastnosti sa vo všeobecnosti dajú rozdeliť do dvoch hlavných skupín a to na podľa toho, či sú závislé od počiatočného značenia, alebo nie. Medzi základné vlastnosti správania sa systému, ktoré závisia od počiatočného značkovania patria: dosiahnuteľnosť, ohraničenosť, živosť, reverzibilnosť, pokryteľnosť, stálosť, korektnosť. Ďalšie štrukturálne vlastnosti, ktoré nezávisia od počiatočného značkovania: štrukturálna živosť, kontrolovateľnosť, štrukturálna ohraničenosť, konzervatívnosť, opakovateľnosť, konzistentnosť.

1.11.1 Analýza Petriho sietí

Existuje viacero možností, ako analyzovať konkrétnu sieť. Medzi ne patrí aj graf dosiahnuteľnosti, vychádzajúci z grafovej reprezentácie stavového priestoru Petriho siete, maticový prístup, dekompozičné a redukčné techniky, teória formálnych jazykov, či ďalšie.

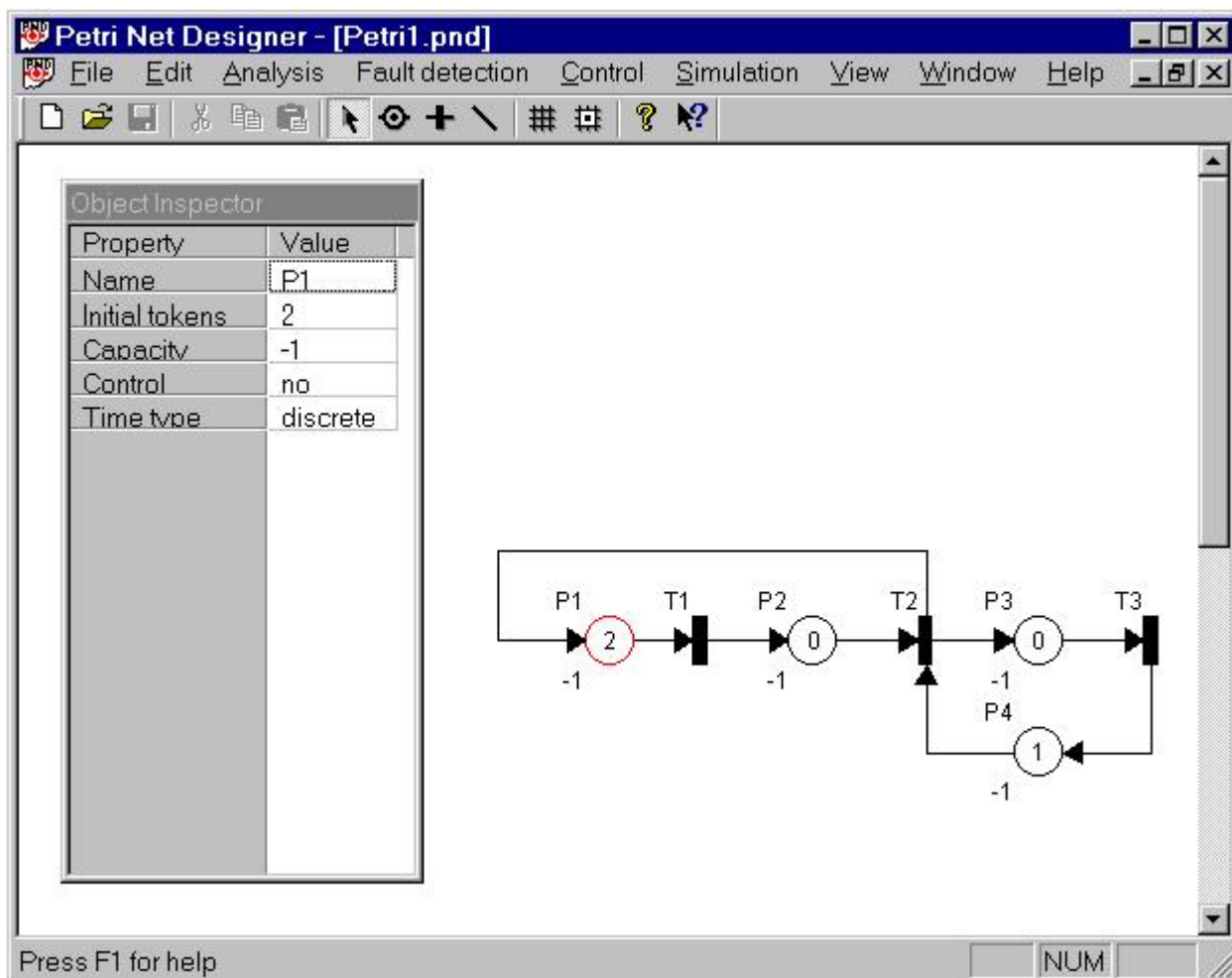
1.11.2 PNDesigner

Petri Nets Designer je aplikačný softvér vyvíjaný na Fakulte informatiky a informačných technológií STU v Bratislave a primárne je určený na pedagogické a výskumné účely. Jeho účelom je modelovanie Petriho sietí a následná analýza pomocou ktorej sa dajú zistiť a skúmať niektoré základné vlastnosti siete: ohraničenosť, živosť, reverzibilitnosť, či analyzovať danú sieť pomocou stromu pokrytia a P, T invariantov. Umožňuje modelovanie jednoduchých, alebo aj zložitejších systémov pomocou intuitívneho rozhrania a grafických prvkov. Ovláda sa štandardným systémom menu, charakteristickým pre okenné aplikácie. Implementačným prostredím programu je Microsoft Visual Studio a je spustiteľný na platformách operačného systému Microsoft Windows. Na obrázku 1.8 možno vidieť obrazovku počas behu programu.

1.11.3 CitectHMI

Citect je spoločnosť, ktorá už viac ako desiatku rokov vyvíja softvér špecializujúci sa na automatizáciu a ovládanie procesov. Medzi jej hlavné produkty patria CitectSCADA, či CitectHMI (obrázok 1.9). Citect HMI je programový balík určený pre vývoj a tvorbu vizualizačných systémov na platforme Microsoft Windows, vychádzajúci zo systému CitectSCADA. Medzi jeho najcharakteristickejšie vlastnosti patrí:

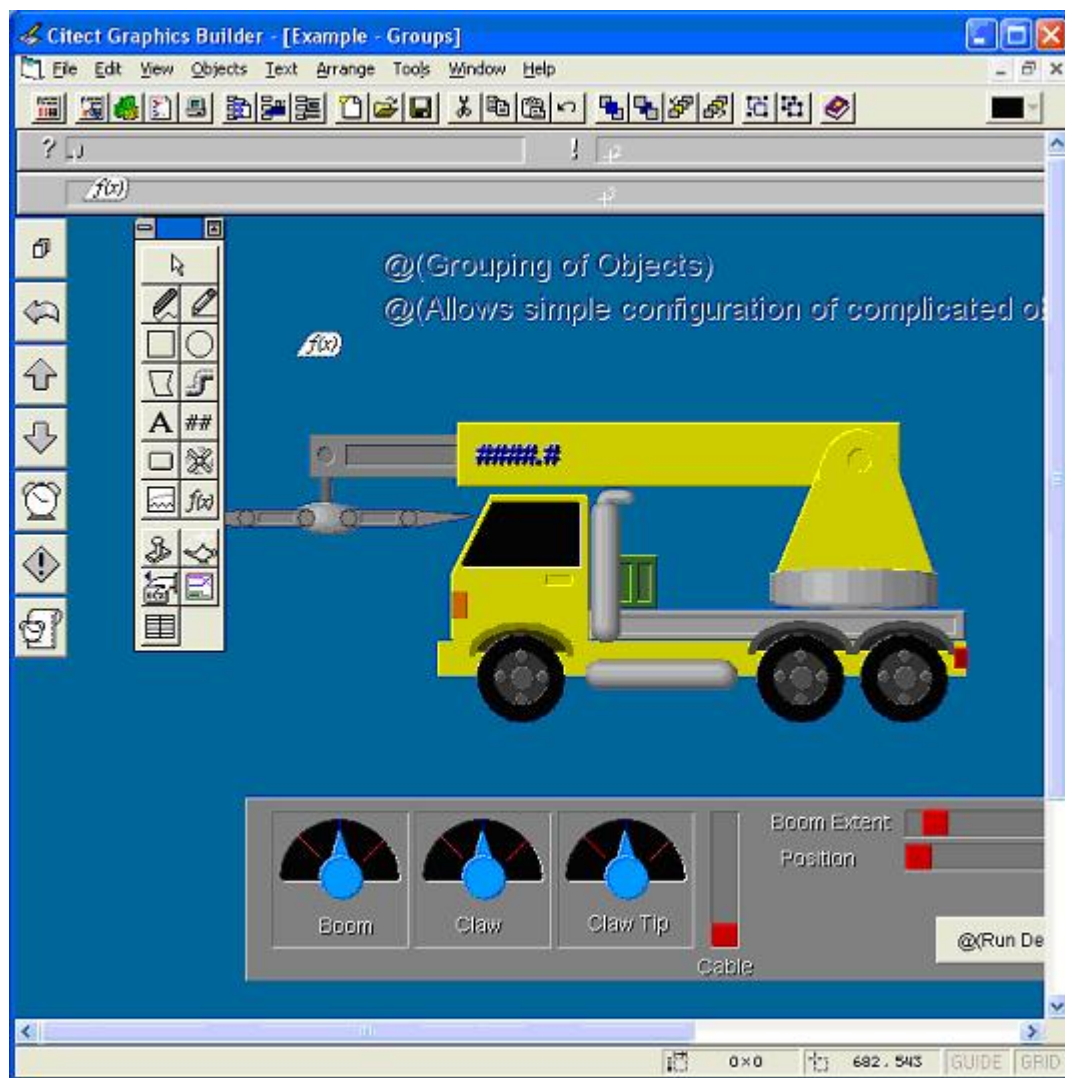
- Pružnosť – jednotlivé stupne vizualizačného systému je možné meniť počas života aplikácie pomocou sprievodcov a zmenou parametrov vo formulároch editora projektu.
- Redundancia – zabudovaná podpora pre redundantnú LAN, či redundanciu súborového serveru.
- Otvorenosť – z tohto hľadiska patrí Citect k naotvorenejším produktom v oblasti SCADA systémov na trhu.



Obrázok 1.8: Petri Net Designer

- Cicode – vlastný programovací jazyk používaný v CitectSCADA. Štruktúra a syntax tohto jazyka je veľmi podobná programovaciemu jazyku Pascal. Medzi hlavné rozdiely patrí absencia smerníkov. Pomocou vstavaných funkcií je možné jednoducho implementovať niektoré zložitejšie jazykové konštrukcie ako súbeh procesov a podobne.

Samotné grafické rozhranie sa vyznačuje svojou rýchlosťou, podporou bežných grafických formátov či šablón a systém umožňuje použitie neobmedze-

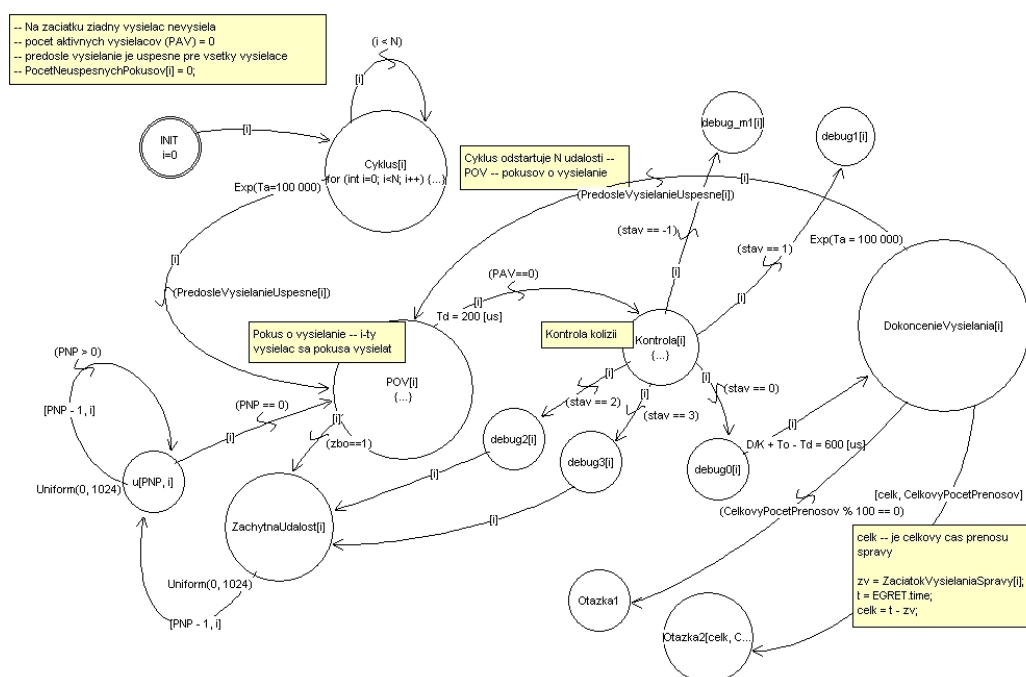


Obrázok 1.9: Časť vývojového prostredia Citect

ného počtu okien a veľký počet animačných prvkov v jednom okne. Obsahuje tiež preddefinovanú knižnicu symbolov parametrizovateľných grafických objektov a okien s možnosťou tvorby vlastných knižníc.

1.12 Grafy udalostí

Ďalším formalizmom, ktorý sme skúmali popri Petriho sieťach, sú grafy udalostí v podobe nástroja *EGRET* obrázok 1.10, C# knižnice *SimuLib* a C knižnice *tiny*. Jedná sa o orientované grafy, kde vrcholy (krúžky) predstavujú udalosti a hrany predstavujú ich plánovanie v čase. Plánovanie ďalších udalostí môže byť podmienené, t.j. aby sa vykonala ďalšia udalosť musí byť splnená daná podmienka. Označuje sa to vlnovkou pri príslušnej hrane (spolu s podmienkou) a ďalej, udalosti môžu byť parametrizovateľné (nad šípku je v hranatých zátvorkách zoznam parametrov, ktoré sa predávajú udalosti). Keď nastane udalosť, vykoná sa zadaný JavaScriptový kód prí-



Obrázok 1.10: EGRET – Graf udalostí

slušnej udalosti. Po jeho dobehnutí sa plánujú udalosti do ktorých smerujú šípky. Plánovanie spočíva v tom, že sa nová udalosť zaradi do kalendára udalostí, podľa zadefinovaného časového oneskorenia, ktoré môže byť dané nahodným rozdelením (s parametrami) alebo s konštantným časovým oneskorením. Pridanie udalosti do kalendára spôsobí jeho usporiadanie podľa

času. Ak sú udalosti plánované na rovnaký čas, rozhoduje sa o ich poradí podľa priority. Podobnú výpočtovú silu ako *EGRET* majú knižnice *SimuLib* a *tiny*, avšak bez názornej vizualizácie.

Kapitola 2

Špecifikácia požiadaviek

Základnou požiadavkou zadania je vytvorenie softvérového produktu implementujúceho niektoré špecifické algoritmy pre riadenie dopravy a ich následná vizualizácia. U nej je vhodné dbať najmä na jednoduchosť použitia, pri zachovaní intuitívnosti ovládania. V tejto časti uvedieme špecifikáciu požiadaviek z hľadiska vizualizácie AGV.

Systém by mal umožniť:

1. Navrhnuť trajektórie, dráhy, cesty a uzly po ktorých sa budú vozidlá pohybovať.
2. Uzly predstavujú zastávky, v ktorých sa vozíky určitý čas zdržia, pričom tento čas bude možné zadávať (konfigurovať, meniť dynamicky v závislosti od typu činnosti).
3. Zastávky môžu byť napríklad prekladiská, parkovacie miesta, čerpacie stanice a pod. K zastávke je možné priradiť textový popis podľa jej účelu.
4. Dráhy budú mať tvar orientovaného grafu s možnosťou zakrivenia hrán ako aj pomenovaním hrán (napríklad názvami ulíc) a ohodnotením hrán podľa ich dĺžky.
5. Bolo by vhodné, keby graf mohol byť priestorový (nie len rovinný), čo by umožnilo napríklad vizualizáciu spomínaných laboratórnych vozíkov pohybujúcich sa vo viacpodlažnej budove.

6. S tretím rozmerom prichádza prirodzene požiadavka približovania a natáčania scény (prototyp nebude obsahovať trojrozmerné scény).
7. Vozíky budú reprezentované jednoduchými geometrickými objektami (kruh, štvorec, resp. ich kompozícia), odlíšiteľných farebne, s možnosťou priradenia identifikátoru k vozíku.
8. K vlastnostiam vozíka bude ďalej patriť aktuálna rýchlosť a priestorová pozícia (poprípade zrýchlenie, alebo kód dynamicky upravujúci rýchlosť na základe vonkajších podnetov). Vozík bude vedieť svoju trasu (t.j. kadiaľ má ísť) a bude obsahovať kód, ako túto trasu dynamicky meniť na základe podnetov zvonka. Trasu vozíkom bude vypočítavať aj optimalizačné jadro programu na základe algoritmu *Free Windows*.
9. Spracovanie vonkajších podnetov môže byť realizované preposielaním správ.

2.1 Špecifikácia funkcií systému

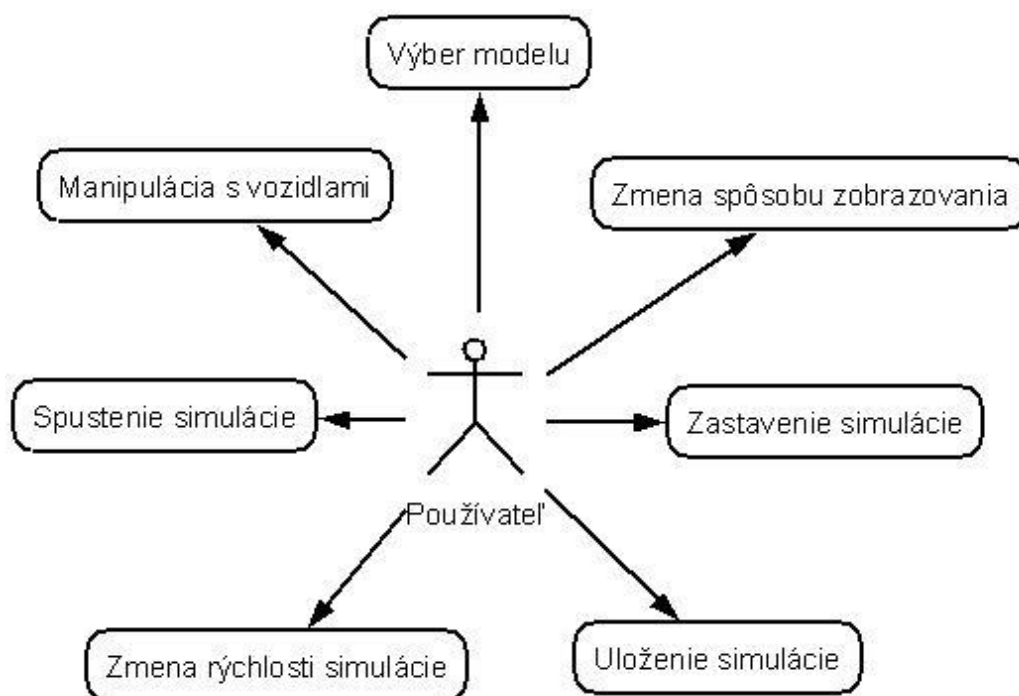
Na obrázku 2.1 je zobrazený diagram prípadov použitia, ktorý predstavuje prehľadný spôsob zobrazenia hlavnej funkcionality navrhovaného programového systému vzhľadom na koncového používateľa v grafickej podobe. Po ňom nasleduje slovný popis a podrobnejšie vysvetlenie jednotlivých funkcií. Niektoré triviálne funkcie a činnosti kvôli väčšej prehľadnosti do diagramu zahrnuté neboli.

Funkcia 1: Výber modelu

Používateľ si po spustení simulátora, alebo aj počas jeho behu, zvolí model pozostávajúci z uzlov, dráh a vozidiel s už definovaným cestovným poriadkom. S týmto modelom môže ďalej pracovať - manipulovať s vozidlami, spúšťať a pozastavovať simuláciu...

Funkcia 2: Spustenie simulácie

Tesne po výbere modelu a po prípadnej manipulácii s vozidlami by mal používateľ po aktivovaní príslušného ovládacieho prvku spustiť simuláciu. Tá pozostáva zo zobrazovania pohybu vozidiel po existujúcich cestách v čase.



Obrázok 2.1: Model prípadov použitia

Funkcia 3: Zastavenie simulácie

Počas behu simulácie je niekedy nutné nielen spomaliť simulačný čas, ale ho aj úplne zastaviť. Typickým príkladom takejto situácie je okamih, kedy používateľ zadáva cieľ cesty viacerým vozidlám súčasne a je podstatný je práve súbežný štart z príslušných zdrojových uzlov.

Funkcia 4: Zmena rýchlosti simulácie

Simulácia vozidiel v modelovanom systéme môže prebiehať v reálnom čase, ale pre účely ďalšieho skúmania je vhodné umožniť zrýchľovanie alebo spomaľovanie simulačného času. Táto zmena neovplyvní rýchlosť samotných vozidiel.

Funkcia 5: Manipulácia s vozidlami

V modelovanom systéme sa pohybuje viacero rozlíšiteľných vozidiel podľa svojich cestovných poriadkov. Používateľ má mať možnosť ľubovoľné z týchto vozidiel odstrániť, či pridať nové. Takisto by mala existovať možnosť meniť cieľ cesty, pre jednotlivé vozidlá.

Funkcia 6: Zmena spôsobu zobrazovania

Vizualizácia modelu a celej simulácie musí umožniť používateľovi zvoliť si úroveň zobrazovaných podrobností a taktiež zmeniť zobrazenie modelu medzi klasickým orientovaným grafom a zobrazením časových okien.

Funkcia 7: Uloženie simulácie

Keďže simulácia môže pozostávať z veľkého počtu vozidiel, ktorých počiatočné a cieľové stanice treba zadávať ručne, je vhodné umožniť používateľovi uložiť tieto parametre pre neskoršie znovupoužitie.

Kapitola 3

Hrubý návrh riešenia

Programový systém by mal pozostávať z dvoch častí. Tou prvou je program implementujúci samotné algoritmy na nájdenie najkratšej možnej cesty v grafe bez narušenia už existujúcich cestovných plánov. Algoritmom na to použitým bude metóda *Free Windows*, prípadne jej vylepšujúce modifikácie. Tento program nebude obsahovať žiadne používateľské rozhranie, na to bude slúžiť druhý program zabezpečujúci celú komunikáciu s používateľom, zobrazovanie simulácie, postaveným na technológií OpenGL.

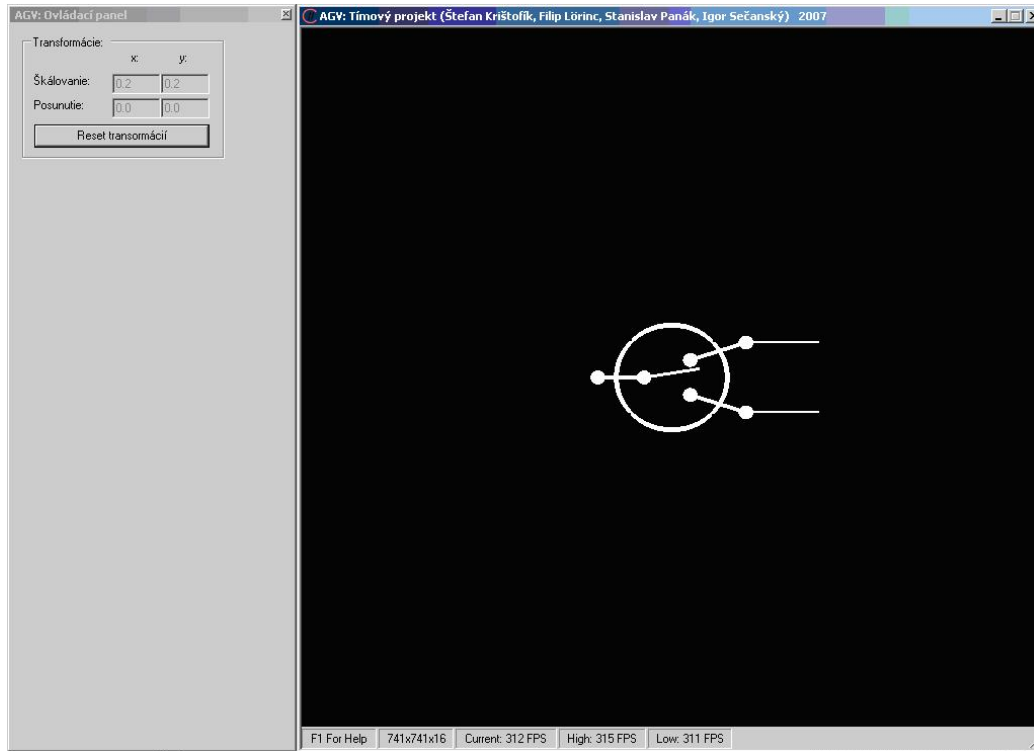
3.1 Cyklus programu

Postupnosť ako budú vykonávané jednotlivé činnosti kombinované s interakciou samotného používateľa s rozhraním znázorňuje diagram na obrázku 3.3. Zo systémového pohľadu sú toto hlavné funkcie pre implementáciu.

3.2 Vizualizácia

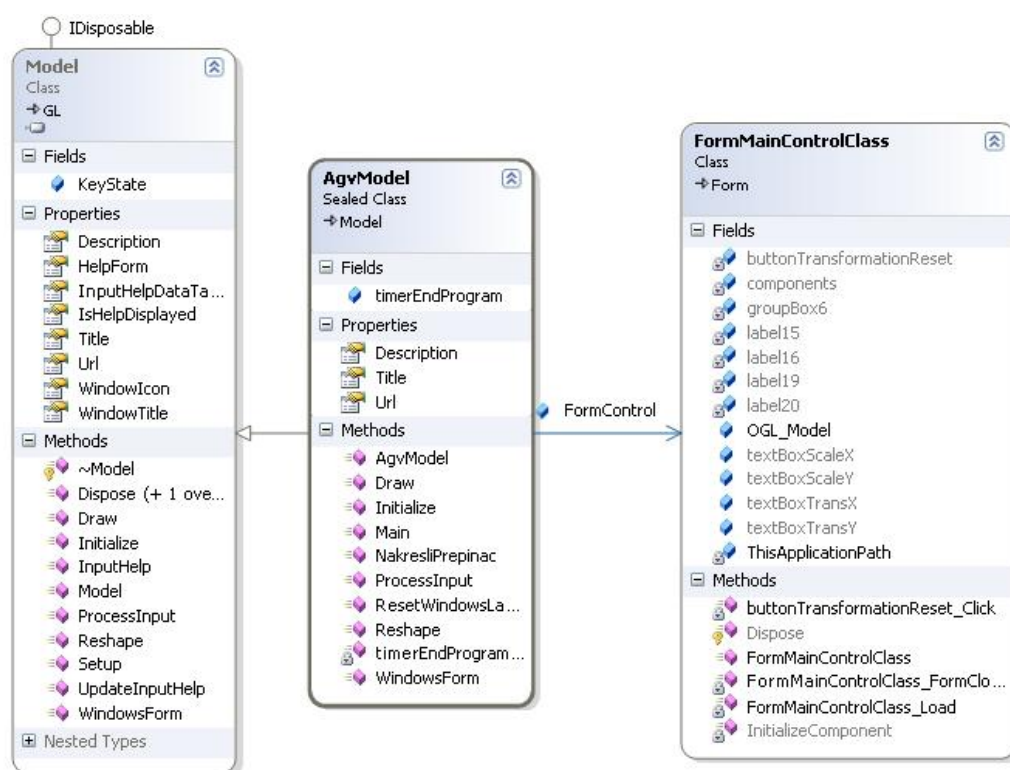
Prototyp je v počiatočných fázach vývoja. Zatiaľ je hotový vizualizačný framework 3.1, kde je panel, v ktorom budú umiestnené ovládacie prvky aplikácie, prepojený s CsGL oknom, v ktorom bude prebiehať vykreslenie scény. Scéna reaguje na zoom a posunutie. Nástroj *Egret* môže byť použitý dvomi spôsobmi: na modelovanie udalostí, aj na nakreslenie topológie trajektorií, kadiaľ budú chodiť vozidlá. Jeho výstupom je XML súbor, ktorý môže byť načítaný do našej aplikácie, ktorá zobrazí identický orientovaný graf. Na uložený súbor z *Egretu* bude zavesený objekt *FileSystemWatcher*, pomocou

ktorého si budeme udržiavať jeho aktuálnu verziu. Nevýhodou *Egretu* je, že nedokáže zobrazovať geometrické objekty, ktoré by sa spojito presúvali po grafe. Preto si túto funkcionality musíme doprogramovať sami. Navrhujeme

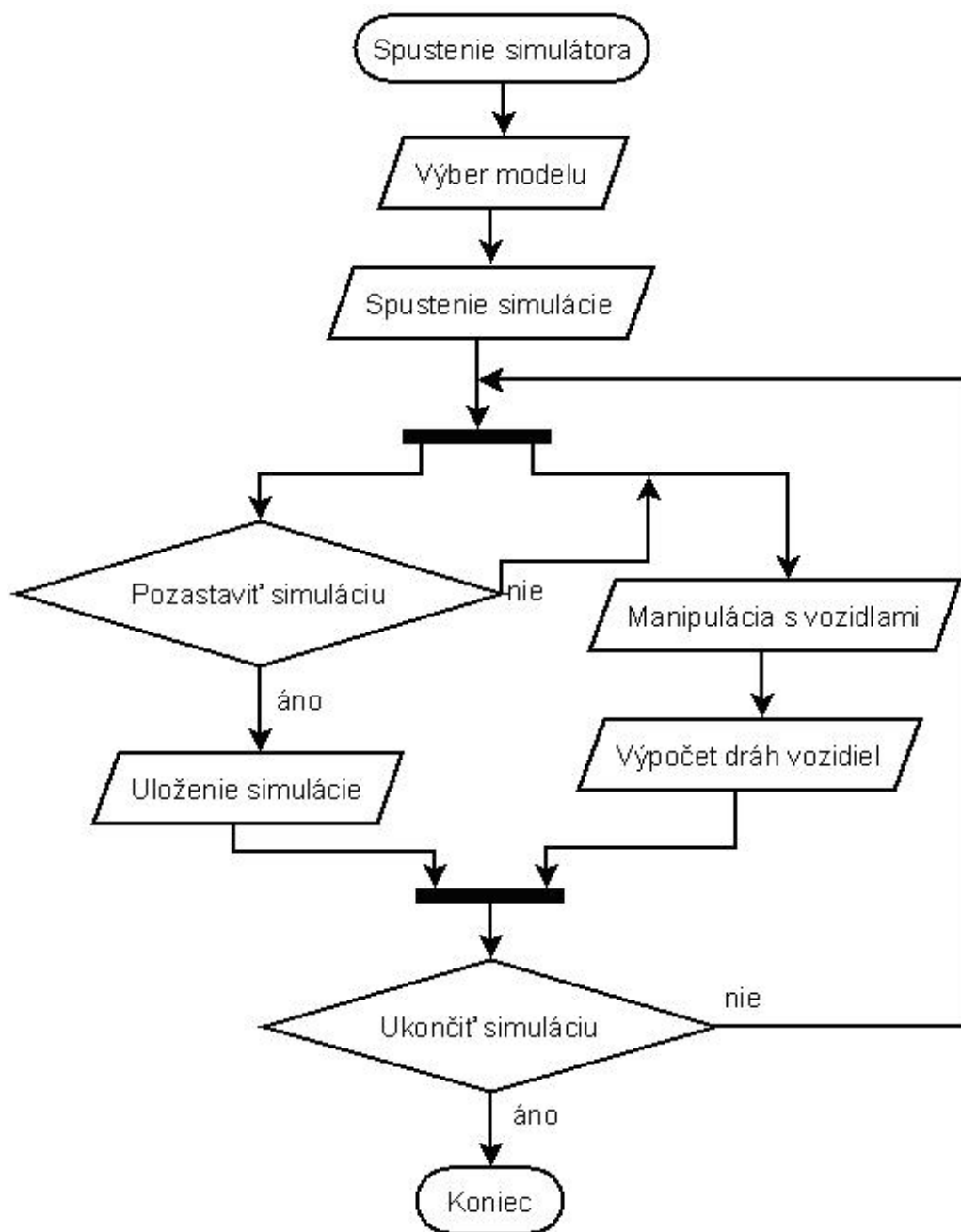


Obrázok 3.1: Vizualizačný framework

využiť plánovacie funkcie knižnice *SimuLib*. UML diagram tried vizualizačného frameworku vidno na obrázku 3.2



Obrázok 3.2: Diagram tried vizualizačného frameworku



Obrázok 3.3: Životný cyklus programu

Literatúra

- [1] Chang W. Kim, J. M. A. Tanchoco: *Conflict-free shortest-time bidirectional AGV routing*, Int. journal Prod. Res. 1991, Vol. 29, No. 12 2377-2391
- [2] Branislav Hruz: *Solution of the Manufacturing Transport Control Using Petri Nets*, IEEE Conference on Systems, Man and Cybernetics, Washington DC, October 2003
- [3] Qiu Ling, Hsu Wen-Jing: *Scheduling and Routing Algorithms for AGVs: a Survey*, 12 October 1999, Technical Report: CAIS-TR-99-26, Centre for Advanced Information Systems, Nanyang Technological University
- [4] http://en.wikipedia.org/wiki/Automated_Guided_Vehicle
- [5] Yoonho Seo, Pius K. Egbelu: *Integrated manufacturing planning for an AGV-based FMS*, Eslevier, International journal of production economics, 1999
- [6] Murata, T., *Petri Nets: Properties, Analysis and Applications* an invited survey paper, Proceedings of the IEEE, Vol.77, No.4 pp.541-580, April, 1989.
- [7] Bielíková, M.: *Ako úspešne vyriešiť projekt*, 1. vyd. Bratislava 2000. 158 s. ISBN 80-227-1329-5

Dodatok

A Metóda Free Windows