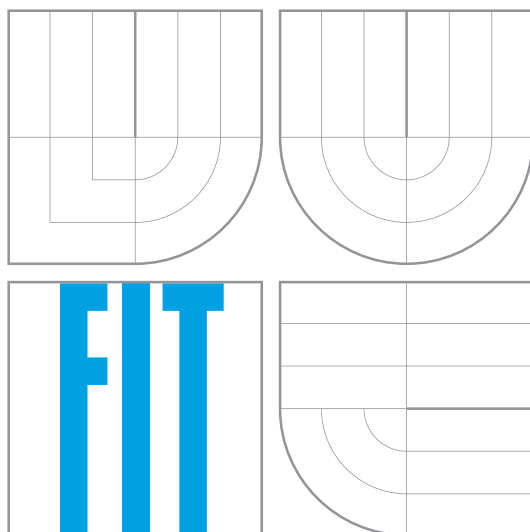


FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



1. Diskrétní simulátor řízený událostmi IMS projekt

Vypracovali: Lenka Jalůvková (xjaluv02), Jiří Pícek (xpícek01)
Dne: 5. prosince 2014

1 Úvod

V této práci je řešena implementace diskrétního simulátoru pro modelování ([4], slajd 119) založeného na kalendáři událostí ([4], slajd 173). Chování tohoto simulátoru je předvedeno na dvou vybraných příkladech z democvičení předmětu IMS, vybrali jsme příklad Učebna a Kravín ([2], slajd 19 a 22). Smyslem experimentu je demonstrovat, že náš simulační nástroj `libsim` odpovídá již existujícím simulátorům.

1.1 Řešitelé a zdroje informací

Projekt vypracoval tým ve složení Lenka Jalůvková a Jiří Pícek. Využili jsme znalosti nabyté na přednáškách a democvičeních předmětu IMS. Problém jsme také prokonzultovali s doktorem Martinem Hrubým.

1.2 Experimentální ověřování validity modelu

Validita vybraných modelů byla ověřena za využití modelů v SIMLIBU odprezentovaných v rámci democvičení předmětu IMS ([3] `kravin.cpp`, `ucebna.cpp`).

Výsledné modely byly testovány na školním serveru `merlin.fit.vutbr.cz`, systém CentOS a také na notebooku se systémem ArchLinux.

2 Rozbor tématu a použitých metod/technologií

Diskrétní simulátor modeluje systém jako diskrétní (nespojitou) posloupnost událostí v čase. Diskrétní simulace je tedy opakem simulace spojité, která kontinuálně zaznamenává dynamiku systému v čase. Spojitá simulace může být také označena jako simulace založená na činnostech. Čas je rozdělen na malé intervaly a stav systému je aktualizován na základě množiny činností, které se odehrávají v daném časovém intervalu. Protože diskrétní simulátor nemusí zpracovávat každý časový interval, mohou běžet mnohem rychleji než odpovídající spojité simulátory [1].

2.1 Použité postupy

Diskrétní simulátor je implementován v jazyce C++, který nám umožnil objektový vývoj. Programování nám zjednodušilo využití tříd, které k sobě svazují data a funkce nad nimi.

2.2 Původ použitých metod/technologií

Při tvorbě projektu nám byla inspirací knihovna SIMLIB, která umožňuje diskrétní i spojitou simulaci, dále demonstrační cvičení a přednášky předmětu IMS.

3 Koncepce

Hlavní komponentou diskrétního simulátoru je kalendář událostí, do kterého přidáváme nebo vyjímáme záznamy. Kalendář událostí je uspořádaná datová struktura uchovávající aktivační záznamy budoucích událostí. Každá naplánovaná budoucí událost *next event* má v kalendáři

záznam obsahující položky $[(acttime_i, priority_i, event_i), \dots]$. Kalendář umožňuje výběr prvního záznamu s nejnižším aktivačním časem a vkládání/rušení aktivačních záznamů.

Princip kalendáře událostí:

```
Inicializace kalendare udalosti a modelu
while(kalendar je neprazdny){
    Vyjmi prvni aktivacni zaznam (AZ) z kalendare
    if (acttime > T_END)
        break; Ukonceni cyklu
    Nastav cas na aktivacni cas acttime v AZ
    Proved popis chovani udalosti event v AZ
}
cas = T_END; konec simulace
```

4 Architektura simulačního modelu/simulátoru

Prvky simulátoru jsou implementovány v pěti hlavních částech (`event.h`, `gen.h`, `libsim.h`, `stat.h` a `store.h`), které tvoří prvky našeho simulátoru `libsim`. Tento simulátor je dále využíván v modelech kravin a učebna, které jsou v souborech `kravin.cpp` a `ucebna.cpp`. Níže jsou jednotlivé části blíže rozebrány.

4.1 Kalendář událostí

Kalendář událostí se nachází v souboru `libsim.h`. Fronta událostí pro kalendář je implementována jako multimapa z knihovny STL (Standard Template Library), která jednotlivé prvky seřadí dle hodnot klíčů, v našem případě časových záznamů. Pokud jsou klíče shodné, jejich pořadí není definováno, to však u simulace není podstatné, protože události se mají provádět ve stejný okamžik, takže nelze rozhodnout, která má přijít dříve.

Hlavní smyčka kalendáře je implementována podle algoritmu výše REFKAL v mírně pozmeněném pořadí. Nejprve proběhne kontrola časové zarážky, dále se nastaví čas na čas aktivace aktuální události a poté se vybere událost a zavolá se její provedení. Kalendář se inicializuje pomocí funkce `init`, které se předává počáteční a koncový čas simulace. Tyto časy je možné kdykoli v simulaci přechíst z proměnných `TIME` a `TIME_END`.

4.2 Události

Další důležitou částí je soubor `event.h` implementující události v simulaci. Události defaultně nemají žádné chování, to se musí vytvořit až v simulaci. Podporují aktivaci v aktuálním čase nebo za zadanou dobu.

4.3 Sklad

Sklad je implementován v souboru `store.h`, který obsahuje základní operace se skladem, a to zabránění skladu (metoda `Enter`) a jeho uvolnění (metoda `Leave`). Dále obsahuje funkce pro zjištění, zda je sklad prázdný (metoda `Empty`) či plný (metoda `Full`).

4.3.1 Zařízení

Při použití skladu s defaultními hodnotami parametrů se chová jako zařízení. Jedná se o velikost skladu u konstruktoru a počet zabíraných nebo uvolňovaných míst u `Enter` a `Leave`.

4.4 Generátor

Generátory jsou implementovány v `gen.h`. Obsahuje pseudonáhodný generátor čísel z intervalu $(0,1)$ a dále generátory s exponenciálním, s normálním a rovnoměrným rozložením.

4.4.1 Pseudonáhodný generátor

Pseudonáhodný generátor jsme implementovali ve funkci `Random`, nejprve jsme se inspirovali z opory lineárním kongruentním generátorem (OPORA str. 18-20), avšak generoval příliš malá čísla i s různým nastavením parametrů, čímž nepříznivě ovlivnil výsledky simulací. Nakonec jsme jej implementovali za využití standardní C funkce `rand`.

4.4.2 Generátor s exponenciálním rozložením

Generátor s exponenciálním rozložením je implementován ve funkci `Exponential`. V tomto případě jsme se inspirovali knihovnou SIMLIB. Pro výpočet využívá výše popsany pseudonáhodný generátor.

4.4.3 Generátor s normálním rozložením

Funkce `Uniform` obsahuje generátor s normálním rozložením. Funkci jsme implementovali dle principu (<http://stackoverflow.com/questions/2325472/generate-random-numbers-following-a-normal-distribution-in-c-c>).

4.4.4 Generátor s rovnoměrným rozložením

Generátor s rovnoměrným rozložením je implementován ve funkci `Uniform`. Tuto funkci jsme implementovali na základě předchozích znalostí.

4.5 Statistiky

V souboru `stat.h` jsou implementovány statistiky, jejichž výstup následně můžeme vypsat po dokončení simulace. Statistické výpisy jsou prováděny pouze pro sklad.

Sledované položky skladu jsou celkový počet požadavků o zabrání skladu, počet požadavků, které zaberou sklad bez čekání ve frontě a statistiky nad frontou u skladu. Sledují se příchozí požadavky ve frontě, dále její maximální, průměrná a aktuální délka a minimální, maximální a průměrný čas čekání.

Funkce statistik se volají v metodách skladu v určitých případech. Pokud se do skladu vstoupí bez čekání ve frontě, zavolá se metoda `EnterNonqueued` ve které se zvýší počet požadavků o sklad a počet požadavků, které prošly bez fronty. Pokud se vstoupí do fronty, tak se uchová aktuální čas, kdy se vstupuje do fronty, zkontroluje se délka fronty pro uchování maximální délky fronty a zvýší se počet požadavků o sklad. V případě opouštění fronty se dopočítávají minimální, maximální čekací čas a počítá se celkový čas strávený ve frontě, který se při výpisu dále přepočítá na průměrný čas čekání ve frontě.

5 Podstata simulačních experimentů a jejich průběh

Činnost simulátoru byla ověřena na dvou modelech z demonstračních cvičení, a to příklad učebna a kravín.

5.1 Postup experimentování

V jednotlivých modelech jsme nejprve nastavili hodnoty parametrů podle příslušných zadání, následně jsme je změnili na základě výsledků. Nakonec jsme je nastavili stejně jako v případě modelů implementovaných pomocí knihovny SIMLIB ([3]).

5.2 Experimenty

Prováděli jsme experimenty nad oběma příklady (Učebna, Kravín) převzatými z demonstračních cvičení předmětu IMS.

5.2.1 Kravín

První experiment je prováděn při nastavení hodnot shodným se zadáním. V kravíně je 100 krav, 5 dojiček, 1 nakládací rampa a 2 auta. Auto může naložit 20 konvic. Experiment je prováděn v simulačním čase 0 – 200 hodin.

```
+-----+
| Store: Rampa
+-----+
| Celkovy pocet pozadavku: 70
| Pocet pozadavku bez fronty: 1
| Aktualni delka fronty: 1
| Maximalni delka fronty: 1
| Minimalni cas cekani: 49.3872
| Maximalni cas cekani: 205.435
| Prumerny cas cekani: 114.588
+-----+
+-----+
| Store: Dojicky
+-----+
| Celkovy pocet pozadavku: 1380
| Pocet pozadavku bez fronty: 1380
| Aktualni delka fronty: 0
| Maximalni delka fronty: 0
| Minimalni cas cekani: -1
| Maximalni cas cekani: 0
| Prumerny cas cekani: -nan
+-----+
```

Obrázek 1: Experiment s nastavením dle zadání

Z výsledku na obrázku 1 vidíme, že dojičky nejsou plně využity. Proto pro další experiment snížíme počet dojiček na 3, stejně jako v simulaci příkladu SIMLIB ([3]).

```

+-----+
| Store: Rampa
+-----+
| Celkovy pocet pozadavku: 64
| Pocet pozadavku bez fronty: 1
| Aktualni delka fronty: 1
| Maximalni delka fronty: 1
| Minimalni cas cekani: 60.4531
| Maximalni cas cekani: 238.83
| Prumerny cas cekani: 131.109
+-----+

+-----+
| Store: Dojicky
+-----+
| Celkovy pocet pozadavku: 1266
| Pocet pozadavku bez fronty: 1170
| Aktualni delka fronty: 0
| Maximalni delka fronty: 7
| Minimalni cas cekani: 0.00830078
| Maximalni cas cekani: 15.0488
| Prumerny cas cekani: 4.31966
+-----+

```

Obrázek 2: Experiment se sníženým počtem dojiček

Z výsledku na obrázku 2 vidíme, že dojičky jsou lépe využity a že maximálně čekalo 7 krav nejdéle však 15 minut. Dále auto čeká na volnou rampu nejméně hodinu. Naplněné auto náklad převáží maximálně hodinu (30 - 60 minut), proto jsme usoudili, že by mohlo stačit pouze jedno auto, tudíž další experiment je pouze s jedním autem.

Z výsledků na obrázku 3 vidíme, že auto přepravilo méně konvic než v předchozím případě, vyplatily by se dvě auta s dvěma rampama.

6 Shrnutí simulačních experimentů a závěr

U statistik se případné další požadavky sledovaných hodnot dají snadno doplnit, my jsme implementovali pouze základní z nich, které jsme shledali důležitými.

```

+-----+
| Store: Rampa
+-----+
| Celkovy pocet pozadavku: 50
| Pocet pozadavku bez fronty: 50
| Aktualni delka fronty: 0
| Maximalni delka fronty: 0
| Minimalni cas cekani: -1
| Maximalni cas cekani: 0
| Prumerny cas cekani: -nan
+-----+

+-----+
| Store: Dojicky
+-----+
| Celkovy pocet pozadavku: 1323
| Pocet pozadavku bez fronty: 1211
| Aktualni delka fronty: 0
| Maximalni delka fronty: 5
| Minimalni cas cekani: 0.0175781
| Maximalni cas cekani: 19.168
| Prumerny cas cekani: 5.23195
+-----+

```

Obrázek 3: Experiment se sníženým počtem aut

Reference

- [1] *Diskretní simulace*. [online], 2013. URL http://www.simulace.info/index.php/Discrete_event_simulation/cs. [cit. 29.11.2014].
- [2] Hrubý M. *Demonstrační cvičení IMS 1*. [online], 2014. URL <http://perchta.fit.vutbr.cz:8000/vyuka-ims/uploads/1/ims-demo1.pdf>. [cit. 29.11.2014].
- [3] Hrubý M. *Demonstrační příklady IMS 2*. [online], 2014. URL <http://perchta.fit.vutbr.cz:8000/vyuka-ims/uploads/1/ims-simlib-dema.tar.gz>. [cit. 27.11.2014].
- [4] Peringer P. *Přednášky Modelování a simulace*. [online], 2014. URL <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IMS-IT/lectures/IMS.pdf?cid=9983>. [cit. 29.11.2014].