Konstrukcja i integracja mechanizmu monitorów statystycznych na potrzeby symulacji w języku C++

Praca magisterska

Marcin Malich

Wojskowa Akademia Techniczna
Wydział Cybernetyki

Warszawa, 2011

Plan prezentacji

1. Wprowadzenie
2. Monitory statystyczne
   - Modelowanie i symulacja
   - Rola i cechy monitorów
   - Rodzaje i budowa monitorów
3. Parametry i charakterystyki
4. Możliwości wizualizacji
5. Biblioteka sim::mon
   - Funkcje i możliwości
   - Implementacja
   - Budowanie i testowanie
   - Korzystanie z biblioteki
6. Symulacja SMO
   - Budowa programu
   - Możliwości programu
7. Podsumowanie

Cel pracy

Cel pracy:

- Konstrukcja monitorów statystycznych w C++
- Analiza i przegląd parametrów i charakterystyk
- Analiza i przegląd możliwości wizualizacji
- Biblioteka *sim::mom*

Modelowanie i symulacja

- System, model, symulacja
- Znaczenie monitorów w symulacji

Rola monitorów

- Obserwowanie i śledzenie zmian
- Gromadzenie danych
- Wyznaczanie statystyk oraz charakterystyk
- Wizualizacja wyników
- Narzędzie do debugowania

Cechy monitorów

- Prosta instalacja
- Transparentność
- Separacja i niezależność
- Szybkość i elastyczność

Rodzaje i budowa monitorów

Rodzaje monitorów:

- zwykłe
- ważone czasem

Konstrukcja monitorów:

- modułowa
- monolityczna

Parametry i charakterystyki

Statystyka opisowa:

- Miary położenia
- Miary zmienności
- Miary asymetrii
- Miary koncentracji

Inne parametry:

- Liczebność, Minimum i Maksimum
- Średnia krocząca
- Histogram
- Wskaźnik struktury

Możliwości wizualizacji

Biblioteki i toolkity:

- wxFreeChart (*wxWidgets*)
- Qwt (*Qt*)

Pakiety i narzędzia:

- gnuplot
- Środowisko R
- MATLAB

Biblioteka *sim::mon*

- Implementacja monitorów w C++
- *Template-based design*
- Multiplatformowość i przenośność
- Licencja MIT

Funkcje i możliwości biblioteki

- Obsługa monitorów zwykłych i ważonych czasem
- Implementacje podstawowych parametrów i statystyk
- Łatwa rozbudowa i rozszerzalność (monitory, statystyki)
- Możliwość monitorowania dowolnych typów elementów
- Powiadamianie o zmianach wartości monitorowanego obiektu
- Możliwość wizualizacji z wykorzystaniem dowolnego softu
- Transparentność, elastyczność i szybkość
- Prosta instalacja oraz wykorzystanie

Zawansowane możliwości

- Statystyki historyczne
- Mechanizm sygnałów
- Powiadamianie o zmianach wartości
- Powiadamianie o zmianach statystyk
- Dowolna obsługa sygnałów (funktory)

Implementacja

- Klasa `BasicMonitor`
- `MonitorTraits`
- `StatFunctorTraits`
- `DefaultParams`
- Funktory statystyk
- Pomocne konstrukcje (`Monitor` i `TMonitor`, i inne)

Budowanie i testowanie

- Brak procesu budowania (kompilacji)
- Prosta instalacja (header path)
- Testy oparte na *Boost Test Library*

Korzystanie z biblioteki

- Proste użycie biblioteki
- Prosta rozbudowa
- Przykładowe aplikacje i kody źródłowe

Program SMO

- Symulacja systemów kolejkowych
- Realne wykorzystanie biblioteki *sim::mon*
- Wykorzystanie biblioteki *cppsim* i *wxWidgets*
- Napisany w języku C++

Budowa programu

- OOP w C++
- Aplikacja wielowątkowa
  - wątek główny (GUI)
  - wątek procesu symulacji
- Zawansowane, dokowane GUI (*wxAUI*)
- Wizualizacja danych oparta na *wxFreeChart*

Możliwości programu

- Konfiguracja systemu SMO
- Sterowanie procesem symulacji
- Monitorowanie wybranych punktów systemu
- Wyznaczanie parametrów i charakterystyk
- Wizualizacja *on-line* wybranych statystyk

Podsumowanie

- Wszystkie założenia i cele zostały zrealizowane
- Łatwa rozszerzalność biblioteki
- Proste wykorzystanie
- Realne zastosowania

Marcin Malich
me@malcom.pl
web: malcom.pl
xmpp: me@malcom.pl