Studium programowania wielowersyjnego

P. Gawkowski, M. Wojciechowski, J. Sosnowski*
Instytut Informatyki, Politechnika Warszawska,
ul. Nowowiejska 15/19, Warszawa 00-665

Streszczenie

W pracy przedstawiono problem jakości i niezawodności w programowaniu wielowersyjnym. Rozważania oparto na badaniach symulacyjnych dla dwu projektów z zakresu sterowania w czasie rzeczywistym.

1. Wprowadzenie

W aplikacjach wymagających dużej niezawodności, bezpieczeństwa itp. pozostawione błędy w oprogramowaniu mogą być zagrożeniem życia człowieka, źródłem poważnych awarii i znacznych strat finansowych. Stąd rozwijane są techniki tolerowania błędów. Są one dobrze znane dla błędów sprzętu. Złożoność oprogramowania jest źródłem wielu błędów zarówno implementacyjnych jak i algorytmów. Stąd nie można polegać tylko na testowaniu oprogramowania. Opracowywane oprogramowanie niesie w sobie ryzyko ujawnienia się błędów niezależnie od realizowanych strategii testowania i walidacji. Tolerowanie tych błędów jest trudniejsze niż błędów sprzętowych. Stosuje się tu różne techniki tolerowania błędów bazują one na redundancji kodu. Wyróżnia się tu programowanie wielowersyjne (N-version programming), alternatywne (recovery blocks) oraz techniki hybrydowe będące kompozycją tych dwu podstawowych [1, 3,6,7].

* Praca finansowana z grantu Dziekana Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej

W programowaniu wielowersyjnym zakłada się realizację obliczeń równolegle dla N wersji i wybór wyniku poprzez głosowanie. W programowaniu alternatywnym też istniej wiele wersji oprogramowania z tym, że jest uruchamiana jedna i jej wynik jest poddawany weryfikacji poprzez test akceptacyjny. W przypadku wyniku negatywnego uruchamiana jest kolejna wersja itd. Rozwiązanie to jest efektywne w przypadku opracowania dobrych testów akceptacyjnych, co jest problematyczne dla wielu aplikacji. Stąd też często stosuje się programowanie wielowersyjne.

Praca dotyczy problemów implementacji koncepcji programowania wielowersyjnego. W szczególności pojawia się problem wyboru poprawnego wyniku. Sama idea głosowania jest prosta ale jej praktyczna implementacja nie jest trywialna [2,5]. Problem te jest bardziej złożony dla oprogramowania niż dla sprzętu. Wynika to przede wszystkim z faktu, że w tym drugim przypadku dość łatwo jest zidentyfikować moduły sprawne. Dysponując kilkoma wersjami oprogramowania nie wiemy, która z nich jest lepsza, jakie zawiera błędy itp. Stąd tez podejmowanie decyzji jest trudniejsze. Ponadto w wielu aplikacjach decyzje musza być podejmowane w sposób ciągły (np. sterowanie obiektem). W przypadku algorytmów bazujących na różnych koncepcjach może to prowadzić do efektu pogorszenia jakości obsługi w porównaniu z implementacją jednowersyjną. Oczywiście system wielowersyjny ma znacznie większą niezawodność. Problem ten nie był praktycznie rozpatrywany w literaturze. W niniejszej pracy zostanie on przedstawiony dla dwóch przykładów: sterowanie ruchem obiektu (S1) i śledzenie obiektu (S2). Dla rozpatrywanych problemów opracowano kilka wersji programowych i stworzono system wielowersyjny a następnie tak przygotowane systemy zostały poddane badaniom jakościowym i niezawodnościowym.

2. System sterowania ruchem obiektu (S1)

Analizowany problem dotyczy sterowania kulki spoczywającej na płaszczyźnie przy pomocy manipulacji nachyleniami tejże płaszczyzny. Postawionym przed systemem zadaniem było doprowadzenie kulki z punktu startowego do określonego miejsca na płaszczyźnie przy obecności zmiennych warunków środowiska (zaimplementowano wiatr o zmieniającym się kierunku i sile jako czynnik zakłócający). Sterowanie odbywa się w kolejnych krokach symulacji (co określony kwant czasu). Zaimplementowano niezależnie trzy moduły obliczeniowe pobierające na wejściu aktualne nachylenie płaszczyzny oraz położenie kulki. Na tej podstawie generują one nowe wartości nachylenia płaszczyzny. Proponowane przez moduły obliczeniowe nachylenia poddawane są następnie głosowaniu, w czasie którego wyłaniane jest nachylenie płaszczyzny, które będzie zastosowane przez kolejny kwant czasu symulacji.

Zbadane zostały własności systemu dla trzech algorytmów głosujących [5]. Na ich potrzeby zdefiniowano dodatkową funkcję wyliczającą dystans pomiędzy dwoma propozycjami wyników. Oba składniki propozycji (nowe nachylenie płaszczyzny oraz kierunek tego nachylenia) są w niej traktowane jako współrzędne punktu na płaszczyźnie - dystans pomiędzy propozycjami jest odległością pomiędzy ich reprezentacją punktową. Zaimplementowano trzy strategie głosowania:

- większościowa z regulowanym progiem - wybierana jest najliczniejsza grupa propozycji, w której dla każdej pary dystans jest mniejszy od zadanego parametrem progu. Liczność takiej grupy powinna być większa od połowy liczności wszystkich propozycji. Z grupy takiej wybiera się następnie losowo jedną z propozycji i traktuje jako wynik obliczeń. W przypadku istnienia dwóch jednakowo licznych grup, najpierw wybiera się jedną z nich i dopiero spośród niej losowo jedno z rozwiązań.
- medianowa - wyliczane są odległości pomiędzy wszystkimi trzema propozycjami a następnie odrzucane są dwie, których odległość jest największa itd. - propozycja "środkowa" staje się wynikiem.
- średnia ważona - wynik ustalany jest niezależnie dla każdego ze składników wyniku poprzez uśrednienie propozycji. Każda składowa propozycji ma przy tym wagę wynikającą z dystansu dzielącego ją od pozostałych propozycji (im większy dystans tym mniejszy wpływ propozycji na uśredniony wynik). Zaimplementowano dwie wersje tej strategii głosowania: w pierwszej (W1) dla obydwu składowych propozycji przypisana była taka sama waga - w wersji drugiej (W2) każda ze składowych propozycji posiadała niezależną wagę wynikającą z dystansu wyliczanego niezależnie dla poszczególnych składowych propozycji.

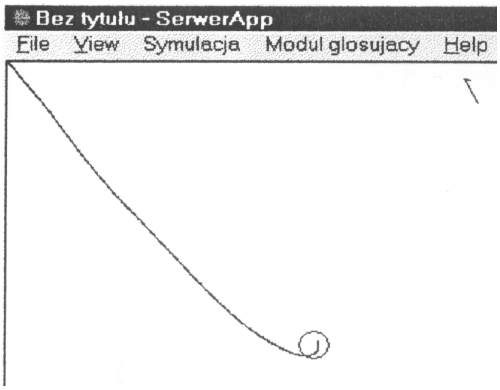
Przeprowadzone eksperymenty odbywały się w identycznych warunkach, tj. przy tych samych parametrach symulowanego świata oraz przy powtarzalnym wietrze. Wyniki najciekawszych eksperymentów zebrano w tabeli 1 oraz przedstawiono na rys. 1 i 2. Zaimplementowany też został moduł oceniający przebieg sterowania. Dostarcza on m.in. takich danych (a prezentowanych w tabeli 1) jak:

- *kroki* - liczba kroków symulacji jaka była potrzebna do wykonania zadania,
- *ilość zmian* - liczba zmian kierunku ruchu kulki w stosunku do kierunku idealnego (wyróżnione są tu jedynie drastyczne zmiany kierunku ruchu kulki - ze zgodnego na przeciwny i odwrotnie),
- *maksymalne odchylenie* - maksymalne odchylenie kulki od trajektorii idealnej - prostej przechodzącej przez punkt początkowy i docelowy,
- *średnie odchylenie* - średnia wartość odchylenia kulki od trajektorii idealnej.

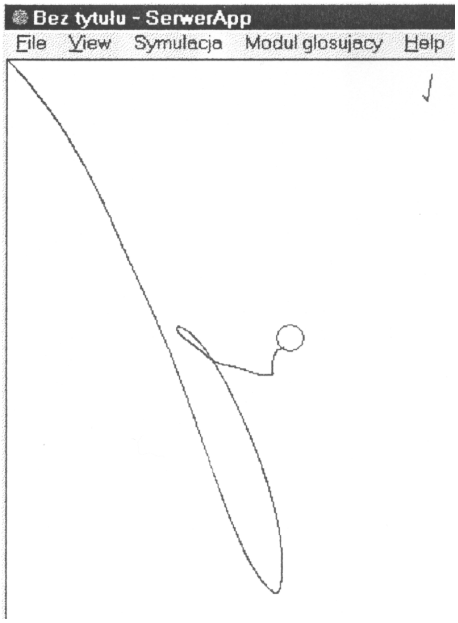
Każdy z niezależnie zaimplementowanych modułów obliczeniowych działając samodzielnie (bez obecności głosowacza) radził sobie bardzo dobrze z

postawionym problemem sterowania. Liczba kroków wahała się w zależności od modułu pomiędzy 1000 a 3000. W każdym jednak przypadku tor kulki był zbliżony do toru idealnego (kulka bez większych odchyleń zdążała do celu).

Rysunki 1 i 2 przedstawiają zrzuty ekranowe z dwóch przykładowych symulacji. W lewym górnym rogu przedstawionych okien znajduje się punkt początkowy (punkt startowy kulki). Okręgiem oznakowano obszar sąsiedztwa punktu docelowego (symulacja kończy się jeśli kulka znajduje się w tym sąsiedztwie przez dostatecznie długi czas). Krzywa łącząca te dwa punkty obrazuje tor ruchu kulki.



Rysunek 1: Tor kulki dla symulacji z głosowaniem większościowym z progiem=0



Rysunek 2: Tor kulki dla symulacji z głosowaniem medianowym

W przypadku stosowania modułów głosujących najlepiej sprawował się algorytm większościowy (patrz tabela 1 oraz rysunki 1 i 2). Testowano go na początku dla wartości progowej równej zero. Powodowało to przypadkowy wybór wyniku (ponieważ nie było grupy propozycji wyniku o liczności 2). Pozostałe algorytmy głosowania dawały dużo gorsze rezultaty (patrz tabela 1). O ile przy rozwiązaniu ze średnią ważoną zrozumiały wydaje się być powód bardzo słabego wysterowania (wynik końcowy jest inny niż jakakolwiek propozycja rozwiązania) o tyle przy algorytmie medianowym, podobnie jak miało to miejsce w głosowaniu większościowym, na wyjście głosowacza podawana jest bezpośrednio jedna z propozycji. Wyniki symulacji pokazują, iż wybór propozycji jako środkowej w oparciu o zaimplementowaną funkcję odległości nie jest dobrym rozwiązaniem. Może to wynikać właśnie z nieadekwatnej do postawionego problemu funkcji odległości wyników (dystansu), ale także może np. świadczyć o niepełnej przypadkowości przy wyborze wyniku w głosowaczu większościowym choć pierwszy powód wydaje się być dużo bardziej prawdopodobny. Świadczyć też o tym może symulacja przy użyciu głosowacza ze średnią ważoną w wersji 2 (W2 - ze zmodyfikowaną funkcją dystansu). W porównaniu z wersją podstawową (W1) tego głosowacza wersja 2-ga pozwoliła na ponad 3-krotne skrócenie liczby kroków symulacji, 4-krotne zmniejszenie średniego odchylenia oraz prawie 10-krotne zmniejszenie maksymalnego odchylenia (patrz tabela 1). Zwiększyła się przy tym liczba zmian kierunku ruchu kulki (liczone są tylko drastyczne zmiany kierunku ruchu - kierunek zgodny albo przeciwny) oraz dynamika tych zmian (liczba zmian / czas symulacji). Niemniej jednak moduł większościowy ciągle zapewniał najlepsze rezultaty. Innym wytłumaczeniem może być charakterystyka poszczególnych modułów - jak się okazało, dwa z nich opierały się na podobnych założeniach - ich wyniki mogły być do siebie zbliżone w wielu sytuacjach - w wyniku czego najczęściej wybierane były propozycje jednego z nich (co faktycznie miało miejsce w czasie eksperymentów z głosowaczem medianowym oraz większościowym z progiem 1.5). Trzeci algorytm odbiegał na tyle od dwóch pozostałych, iż nierównomierne z nich korzystanie prowadziło do pogorszenia efektów symulacji. Przeprowadzono także eksperymenty z udziałem modułu większościowego dla różnych wartości progu (patrz tabela 1). Można zauważyć, że dla małych wartości progu głosowacz losowo wybiera wynik - w przypadku opisywanego problemu daje to bardzo dobre rezultaty. Kiedy próg jest zwiększany dochodzi do sytuacji kiedy pojawia się para propozycji uznanych za zgodne - wyniki takich symulacji są nieco gorsze - przypominają nieco symulacje z udziałem głosowacza medianowego. Wynika to z podobieństw między dwoma algorytmami - ich para propozycji przegłosowuje trzeci algorytm i podobnie jak ma to miejsce w przypadku wyboru mediany wybierane są ich propozycje częściej niż trzeciego algorytmu.

Pomimo widocznego pogorszenia się efektywności sterowania w konfiguracji 3-wersyjnej w stosunku do wersji z jednym modułem obliczeniowym przeprowadzone eksperymenty dowiodły odporności na błędy pierwszej konfiguracji. Każdy z modułów obliczeniowych pracował na oddzielnej maszynie

i komunikował się poprzez TCP/IP z modułem głosującym. W trakcie eksperymentów z głosowaczem ze średnią ważoną doszło do sytuacji zerwania połączenia TCP/IP z jednym z modułów obliczeniowych. Pomimo, że przyjmowane są wówczas dane zerowe jako wynik od "utraconego" klienta udało się dość dobrze wysterować kulkę do celu.

Sposób głosowania	Liczba kroków	Ilość zmian	Maksymalne odchylenie	Średnie odchylenie
Większościowe (próg=0)	720	1	21,8	11,5
Większościowe (próg=1,5)	1141	4	61,7	24,1
Większościowe (próg=2)	745	2	26,1	12,3
Medianowe	3486	8	137,9	42,7
Średnia ważona (W1)	6645	17	218,5	56,1
Średnia ważona (W2)	2015	73	24,9	12,6

Tabela 1. Wyniki eksperymentów dla różnych modułów głosujących.

3. System śledzenia obiektu (S2)

Analizowany problem dotyczy śledzenia obiektu A przez obiekt B. Oba obiekty poruszają się na płaszczyźnie 4000x4000 punktów. Obiekt A porusza się w sposób przypadkowy z zadaną długością skoku i jego częstotliwością. Obiekt śledzący B ma za zadanie zbliżyć się do obiektu A na odległość nie większą niż d. Opracowano trzy algorytmy poruszania się obiektu B. Algorytm A1 jest najszybszy. W każdym kroku tworzona jest prosta łącząca oba obiekty i obiekt B porusza się po tej prostej. W algorytmie A2 obiekt B porusza się tylko wzdłuż jednej osi (x lub y), dla której jest największa odległość do obiektu A. W algorytmie A3 sterowanie obiektem B w dużym zakresie ma charakter przypadkowy (wybierana jest najlepsza z trzech generowanych przypadkowo decyzji).

Opracowane algorytmy wykorzystano w realizowanym systemie 3-wersyjnym z głosowaniem. Rozpatrzono głosowanie według mediany, większościowe i ważone (porównaj rozdz. 2). Przeprowadzono symulacje działania systemu dla tych technik głosowania. Dla każdej techniki brano pod uwagę różną częstotliwość podejmowania decyzji o wysterowaniu obiektu B na podstawie wyniku głosowania. Częstotliwość podejmowania decyzji była określana przez podanie liczby kroków (K) między kolejnymi punktami

decyzyjnymi. Wyniki symulacji zestawiono w tabelach 2-4 dla K=1-50. Dla każdej symulacji przedstawiono liczbę kroków (N), po których obiekt B zbliżył się do obiektu A na odległość nie większą niż zadana wartość d (d<20) oraz czas (T0) po jakim to nastąpiło. Ponadto podano parametr zredukowany T0/N. Dla porównania w tabeli 5 przedstawiono wyniki symulacji dla systemu jednowersyjnego działającego zgodnie z wybranym jednym algorytmem. Dla systemu wielowersyjnego z głosowaniem większościowym i ważonym podano również wyniki w przypadku zakłócania jednej z wersji (symulacja błędu programu i algorytmu). Zakłócenia były dobrze tolerowane przez system wielowersyjny. System wielowersyjny wykazywał dłuższy okres dochodzenia do celu niż najlepszy system jednowersyjny (algorytm A1). Różnica ta jednak nie przekraczała 20%. Na uwagę zasługuje fakt, że różnica ta była największa dla systemów z najczęściej podejmowaną decyzją. Zwiększając odstęp między kolejnymi decyzjami osiągnięto polepszenie czasu dochodzenia do celu (gorszy o mniej niż 10% od optymalnego). W przypadku zakłóceń wyniki były bardziej przypadkowe w funkcji liczby punktów decyzyjnych. Najlepsze wyniki uzyskano dla głosowania ważonego. Dość mały czas dochodzenia wynikał z faktu zakłócania najgorszej wersji.

W tabelach 2-5 podano wyniki uśrednione dla 1000 symulacji odpowiadających różnym sytuacjom ustawień początkowych obiektów A i B.

K	N	T0	N/T0
1	66,7	16,9	3,95
2	68,0	16,4	4,16
3	68,1	15,9	4,29
4	68,9	15,5	4,43
5	68,9	15,5	4,46
6	69,8	15,7	4,43
7	69,8	15,4	4,55
8	71,4	15,5	4,59
9	69,5	15,3	4,54
10	68,3	15,1	4,53
15	70,3	15,3	4,60
20	68,6	15,1	4,55
25	68,7	15,1	4,55
30	69,2	15,1	4,57
35	69,5	15,2	4,59
40	69,6	15,1	4,62
45	69,9	15,2	4,60
50	69,9	15,2	4,60

Tabela 2. Głosowanie medianowe

	Bez zakłócania			Z zakłóceniem		
K	N	T0	N/T0	N	T0	N/T0
1	73,7	17,4	4,2	74,9	17,7	4,2
2	73,4	16,6	4,4			
3	74,3	16,1	4,6			
4	74,7	16,1	4,6			
5	74,9	15,9	4,7	76,9	16,0	4,8
6	74,5	15,9	4,7			
7	75,4	15,9	4,7			
8	74,4	15,9	4,7			
9	73,7	15,6	4,7			
10	75,5	15,7	4,8	80,1	15,8	5,1
15	75,9	15,8	4,8	87,5	16,9	5,2
20	76,0	15,7	4,8	79,7	15,7	5,1
25	76,7	15,8	4,9	89,5	17,0	5,3
30	77,1	15,8	4,9	79,2	15,5	5,1
35	76,3	15,7	4,9	90,4	17,1	5,3
40	76,7	15,8	4,9	80,0	15,6	5,1
45	76,3	15,7	4,9	92,1	17,3	5,3
50	76,2	15,7	4,9	80,2	15,6	5,2

Tabela 3. Głosowanie większościowe

	Bez zakłócania			Z zakłóceniem		
K	N	T0	N/T0	N	T0	N/T0
1	71,0	15,8	4,5	55,5	14,1	3,9
2	69,9	15,7	4,5			
3	69,0	15,5	4,4			
4	70,6	15,8	4,5			
5	69,7	15,5	4,5	54,5	13,9	3,9
6	70,0	15,5	4,5			
7	69,0	15,4	4,5			
8	68,9	15,4	4,5			
9	69,7	15,5	4,5			
10	69,3	15,5	4,5	59,2	15,5	4,1
15	71,4	15,6	4,6	63,9	15,0	4,3
20	71,1	15,7	4,5	57,7	14,2	4,0
25	69,0	15,4	4,5	65,0	15,1	4,3
30	70,6	15,6	4,5	58,8	14,3	4,1
35	71,3	15,7	4,5	64,4	15,1	4,3
40	71,3	15,7	4,5	59,3	15,5	4,1
45	71,0	15,7	4,5	67,7	15,4	4,4
50	70,5	15,6	4,5	58,4	14,3	4,1

Tabela 4. Głosowanie ze średnią ważoną

Algorytm	N	T0	N/T0
A1	60,9	14,5	4,2
A2	80,0	15,3	5,2
A3	92,5	17,2	5,4

Tabela 5. Efektywność algorytmów

4. Podsumowanie

W pracy pokazano efekt pogarszania się jakości obsługi systemu z programowaniem wielowersyjnym w porównaniu z systemem jednowersyjnym. Efekt ten był silniejszy dla problemu S1. W przypadku problemu S2 pogorszenie jakości jest dość ograniczone. Stąd opracowując systemy wielowersyjne należy zwrócić uwagę na ten efekt i przyjąć odpowiednie strategie wyboru wyników, częstości podejmowania decyzji itd. Zagadnienie to się znaczne komplikuje, jeżeli istnieją istotne różnice jakościowe w wykorzystywanych algorytmach. Bardzo pomocne są tu badania eksperymentalne. Potwierdziły to przedstawione przykłady, które opracowano dla potrzeb ilustracji zaobserwowanego efektu. Opisany efekt nie wystąpi lub wystąpi w ograniczonym zakresie, jeżeli stosowane algorytmy są kompatybilne, bazują na tych samych danych wejściowych, tak samo interpretują stan systemu sterowanego itp. Wprowadzanie takich ograniczeń w programowaniu wielowersyjnym może być niekorzystne, gdyż prowadzi do ryzyka błędów skorelowanych [4]. Stąd też istotnym jest zapewnić dużą efektywność każdej wersji oraz stworzyć możliwości łatwego przełączania z jednej wersji na inną przy ograniczeniu spadku efektywności.

Warto nadmienić, że w literaturze opisywano podobny efekt [4], ale dotyczący problemu niezawodności. Mianowicie pokazywano przykłady, gdy niezawodność wersji najlepszej była wyższa od systemu wielowersyjnego. Przy czym w przypadku stosowania wersji o dużej niezawodności ten efekt był mniej widoczny. W rozważaniach tych były istotne uproszczenia, które prowadziły do zbyt pochopnych wniosków.

Bibliografia

[1]. N. Ashrafi, O. Berman, M. Cutler. Optimal Design of Large Software-Systems Using N-Version Porgramming. *IEEE Transactions on Reliability*, Vol. 43, No. 2, pp. 344-350, June 1994

[2]. P. R. Lorczak, A. K. Caglayan, D. E. Eckhardt. A Theoretical Investigation of Generalized Voters For Redundant Systems. *FTCS-19*, pp. 444-451, Chicago 1989

[3]. M. R. Lyu. *Software fault tolerance*. John Wiley, Chichester, 1995

[4]. M. R. Lyu. *Handbook of software reliability engineering*. McGraw Hill, New York, 1996

[5]. B. Parhami. Voting Algorithms. *IEEE Transactions on Reliability*, Vol. 43, No. 4, December 1994, pp. 617-629

[6]. A. Romanovsky. On N-version Programming and Exception Handling. *EWDC-10*, Vienna, pp. 175-179, May 6-7 1999

[7]. T. Tsai. Fault Tolerance via N-Modular Software Redundancy. *FTCS-28*, pp. 201-206, 1998