CZASOPISMO TECHNICZNE
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ

z. 1-I/2007
ISSN 0011-4561
ISSN 1897-631X

KRZYSZTOF SAPIECHA, JOANNA STRUG*, PRZEMYSŁAW MAKSYM**

KREATOR SCENARIUSZY TESTOWYCH DO WALIDACJI REAKTYWNYCH SYSTEMÓW ZAMKNIĘTYCH

THE GENERATOR OF TEST SCENARIOS FOR VALIDATION OF REACTIVE EMBEDDED SYSTEMS

Streszczenie

Podstawową techniką walidacji w przypadku reaktywnych systemów zamkniętych jest symulacja. Wyznaczenie odpowiedniego zbioru scenariuszy testowych, pozwalających na sprawdzenie poprawności funkcjonalnej i czasowej projektowanego systemu, jest zadaniem trudnym i bardzo pracochłonnym. Zatem duże znaczenie praktyczne ma automatyzacja tego procesu. W niniejszym artykule przedstawiono prototypowe narzędzie implementujące oryginalną metodę generacji scenariuszy testowych dla reaktywnych systemów zamkniętych oraz wyniki eksperymentów przeprowadzonych z jego zastosowaniem.

Słowa kluczowe: walidacja, automatyczna generacja scenariuszy testowych, systemy zamknięte

Abstract

Simulation is the key validation technique for reactive embedded systems. Generation of a suitable set of test scenarios for checking functional and temporal correctness of a system is a difficult task. Computer-aided test scenarios generation has thus a practical meaning. This paper presents a tool implementing a method of test scenarios generation for reactive embedded systems and the results of experiments carried out with the help of this tool.

Keywords: validation, automatic test scenarios generation, embedded systems

* Prof. dr hab. inż. Krzysztof Sapiecha, dr inż. Joanna Strug, Katedra Informatyki Technicznej, Wydział Inżynierii Elektrycznej i Komputerowej, Politechnika Krakowska.

** Mgr inż. Przemysław Maksym, dyplomant, Katedra Informatyki Technicznej, Wydział Inżynierii Elektrycznej i Komputerowej, Politechnika Krakowska.

Oznaczenia

GWF – graf wymagań funkcjonalnych
DST – drzewo scenariuszy testowych
RId – identyfikator wymagania funkcjonalnego
CId – identyfikator ograniczenia czasowego
SK_{CId} – ścieżka krytyczna dla ograniczenia CId
SF_{RId} – ścieżka funkcjonalna dla wymagania RId
TId – identyfikator zadania/trybu uruchomieniowego zadania
t_{min} – minimalne opóźnienie czasu zakończenia wykonywania podzbioru zadań
t_{max} – maksymalne opóźnienie czasu zakończenia wykonywania podzbioru zadań

1. Wstęp

Zainteresowanie na świecie tematyką walidacji systemów cyfrowych jest duże, co znajduje odbicie w wielu pracach poświęconych zarówno metodom opartym na symulacji [6, 5, 11, 20, 10, 12], jak i metodom formalnym [2]. Jednak obydwie techniki wciąż borykają się z pewnymi problemami. Metody formalne, ze względu na dużą złożoność i ograniczenie zakresu zastosowania do modeli projektowanych systemów, mogą być stosowane głównie do walidacji programów małych i średnich. Natomiast w przypadku metod symulacyjnych problemem jest wciąż wygenerowanie praktycznego pod względem rozmiaru zbioru przypadków testowych, który umożliwi uzyskanie wiarygodnych wyników walidacji.

Istnieje kilka podejść do generacji przypadków testowych. W [5, 20] do generacji testów funkcjonanych stosowane są narzędzia ATPG (ang. *Automatic Test Pattern Generation*). W [11] posłużono się algorytmem genetycznym SG (ang. *Selfish Gene*), którego celem jest ocena i modyfikacja losowo wygenerowanego zestawu testów w taki sposób, aby ostateczny zbiór testów zapewniał jak najwyższe pokrycie krawędzi w modelu CFSM (ang. *Codesign Finite State Machine*) reprezentującym system. W podejściu zastosowanym w [20] zmodyfikowano algorytmy ATPG, tak aby możliwe było rozpatrywanie tzw. celów walidacyjnych (wyrażeń warunkowych i arytmetycznych, przypisań zmiennych). W trakcie symulacji generowane są środowiska testowe, będące ścieżkami prowadzącymi od jednego z wejść pierwotnych systemu, poprzez określony cel, do pewnego wyjścia systemu. Wszystkie trzy metody są przeznaczone dla systemów realizowanych sprzętowo, dla których dostępny jest opis w postaci kodu w języku VHDL. W [6] skoncentowano się na systemach wbudowanych. Do wyznaczenia zbioru scenariuszy testowych zastosowano tutaj technikę przeszukiwania przestrzeni stanów modelu wymagań funkcjonlanych (ang. *functional requirements model*) otrzymanego na podstawie specyfikacji SCR. W [10] przedstawiono metodę automatycznej generacji „czasowych" przypadków testowych dla systemów, których zachowanie i środowisko są opisane w formie DIEOU-TA (ang. *Deterministic, Input Enabled, Output Urgent TA*), będącego ograniczoną wersją TA (ang. *Timed Automation*) [1]. Do generacji sekwencji testowych stosowany jest weryfikator UPPAAL, który pozwala na wyznaczenie śladów diagnostycznych badających spełnienie pewnych celów walidacyjnych lub kryteriów pokrycia (pokrycia krawędzi (ang. *edge coverage*), stanów (ang. *location coverage*) i przepływu danych (ang. *data-flow coverage*)). W [12] przedstawiono technikę testowania on-line (jednoczesna generacja i zastoswanie testów) prze-

znaczoną do walidacji ograniczeń czasowych w systemach wbudowanych. Także w tej metodzie modelem systemu jest TA. Generacja „czasowych" sekwencji testujących opiera się na idei obliczania podzbioru stanów osiągalnch w modelu TA w wyniku zastosowania określonego pobudzenia (ang. *input*). Powyższa metoda została zaimplementowana w postaci narzędzia TRON.

W generacji testów znajdują także zastosowanie techniki pierwotnie opracowane na potrzeby metod formalnych, głównie sprawdzania modelu (ang. *model checking*) [3]. W [15] do generacji funkcjonalnych testów walidacyjnych dla procesora stosowany jest weryfikator SMV (ang. *Symbolic Model Veryfier*) [13]. Na podstawie specyfikacji procesora w języku ADL (ang. *Archtecture Description Language*) generowany jest model w formie akceptowalnej przez SMV. Następnie definiowane są własności (przeciwne do pożądanych własności systemu), a weryfikator generuje dla każdej z nich kontrprzykład przekształcany na tzw. program testowy składający się z instrukcji procesora. Jednak ze względu na ograniczenia stosowanego narzędzia takie rozwiązanie nie umożliwia opisania procesora z odpowiednim stopniem szczegółowości.

W niniejszym artykule zaprezentowano kreator scenariuszy testowych implementujący oryginalną metodę generacji zredukowanego zbioru scenariuszy testowych do walidacji reaktywnych systemów zamkniętych, której pełny opis znajduje się w [16]. W rozdziale 2, na prostym przykładzie, wyjaśniono podstawy implementowanej metody oraz pokazano przebieg procesu generacji scenariuszy testowych. Motywację zautmatyzowania tego procesu zawarto w rozdz. 3. W rozdziale 4 scharakteryzowano możliwości Kreatora ST automatyzującego przedmiotowy proces generacji. Wyniki eksperymentów wykonanych przy zastosowaniu Kreatora ST zebrano i omówiono w rozdz. 5. Rozdział 6 stanowi podsumowanie artykułu.

2. Generacja scenariuszy testowych

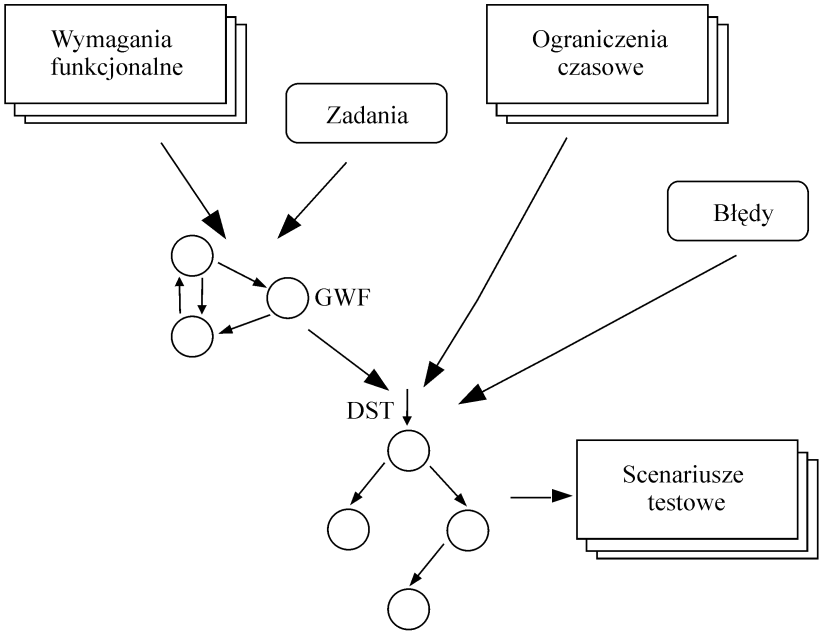
Walidowane systemy, oprócz cech typowych dla systemów reaktywnych [4], charakteryzują się stałością wykonywanych funkcji, jednolitą realizacją i stałym czasem wykonania poszczególnych zadań lub ich trybów uruchomieniowych oraz wyróżnialnym stanem początkowym [19]. Na rycinie 1 pokazano schemat generacji scenariuszy testowych dla tych systemów według metody podanej w [16].

Do zilustrowania przedstawionej metody posłuży prosty system CO2 mierzący stężenie CO_2 w powietrzu. System po włączeniu rozpczyna rejestrację stężenia, a następnie wyświetla na monitorze LCD wartość pomiaru. Pomiar można powtórzyć po zresetowaniu systemu. Wyłączenie systemu kończy jego pracę.

2.1. Specyfikacja wymagań systemowych, zadania i błędy projektowe

Do sformalizowania wymagań funkcjonalnych jest stosowana notacja SCR (ang. *Software Cost Reduction*) [9], która umożliwia opisanie zachowania systemu w kategorii zdarzeń wywołujących zmiany w systemie. Do utworzonej specyfikacji SCR wprowadzane są dane o zadaniach realizujących poszczególne wymagania (dokonanie podziału funkcji na zadania już na tym etapie jest możliwe ze względu na specyfikę systemów zamkniętych). To uzupełnienie rozszerza zasób wiedzy o systemie dostępnej podczas wyznaczania scena-

riuszy. Na poziomie specyfikacji SCR każde z zadań jest rozumiane jako ciąg pewnych czynności (dokładnie określonych dopiero w trakcie projektowania), których wykonywanie rozpoczyna się wraz z pojawieniem się określonych zdarzeń (zdarzenia inicjujące) i kończy wyprowadzeniem wyniku (zdarzenia kończące).



Ryc. 1. Zarys generacji scenariuszy testowych

Fig. 1. Test scenarios generation flow

Specyfikację wymagań funkcjonalnych, w notacji SCR, dla systemu CO2 zawarto w tablicach 1 i 2. Stan systenu jest określony przez dwie zmienne, SYSTEM i LCD, reprezentujące odpowiednio tryby pracy systemu (Wył. i Wł.) i stany monitora LCD (0 – wygaszony, L – gotowość i xSENSOR – wyświetlenie pomiaru). Obydwie tablice zawierają opis zmian wartości poszczególnych zmiennych zachodzących pod wpływem zadanych zdarzeń. Z każdą zmianą wartości wiąże się zrealizowanie pewnego wymagania funkcjonalnego (RId) oraz wykonywanie określonego zadania (TId). Zadanie TS wykonuje operacje związane z włączeniem i wyłączeniem systemu, a zadanie TL realizuje opercje związane z pomiarem i obsługą monitora. Pierwszy wiersz tabl. 1 informuje o tym, że wymaganie R1a, polegające na wykonaniu zadania TS_{ON}, realizuje zmianę stanu zmiennej SYSTEM z Wył. na Wł. pod wpływem zdarzenia SWITCH = ON.

Opracowana metoda dopuszcza definiowanie wartości symbolicznych reprezentujących podzbiór dopuszczalnych wartości dla zmiennych stanu, wejść i wyjść systemu. Dla systemu CO2 zdefiniowano takie wartości dla portu SENSOR i zmiennej (wyjścia) LCD.

Tablica 1

Tabela SCR dla zmiennej SYSTEM

SYSTEM				
Tryb	nowy tryb	Zdarzenie	RId	TId
Wył.	Wł.	@T(SWITCH = ON)	R1a	TS_{ON}
Wł.	Wył.	@T(SWITCH = OFF)	R1b	TS_{OFF}

Tablica 2

Tabela SCR dla zmiennej LCD

LCD									
Tryb	0			L			xSENSOR		
	zdarzenie	RId	TId	zdarzenie	RId	TId	zdarzenie	RId	TId
Wył.	@T(Inmode)	R2a	TL_0	@T(False)	–	–	@T(False)	–	–
Wł.	@T(False)	–	–	@T(Inmode) @T(RESET=1)	R2b R2d	TL_L	@T(SENSOR =xSENSOR	R2c	TL_S

Ograniczenia czasowe zdefiniowane dla systemów zamkniętych uwzględnają ograniczenia nałożone na system i na jego otoczenie [7]. Każde ograniczenie nałożone na system wyznacza okres (podany w postaci przedziału ($t_{min.}$, t_{max})), w którym system musi zakończyć przetwarzanie pewnych danych. Przetworzenie danych polega na wykonaniu określonych zadań, zatem przyjęto założenie, że ograniczenia te są nałożone na czas wykonania przez system pewnego podzbioru zadań.

W tablicy 3 pokazano specyfikację ograniczeń czasowych dla systemu CO2 zapisaną z zastosowaniem rozszerzonej notacji SCR [16]. Zawarto w niej także identyfkator ograniczenia (CId), jego typ, przedział czasu zakończenia wykonywania zadań oraz, w zależności od typu ograniczenia, podzbiór zadań, na które dane ograniczenie jest nałożone (dla typu Z obowiązkowy) lub warunki określające moment rozpoczęcia i zakończenia wykonywania zadań (dla typu P obowiązkowy) [18]. Pierwszy wiersz tabl. 3 wskazuje, że ograniczenie C1 polega na wykonaniu zadań inicjowanych zdarzeniem SWITCH = ON lub RESET = 1 i kończących się pojawieniem się zdarzenia LCD = L.

Tablica 3

Ograniczenia czasowe dla systemu CO2

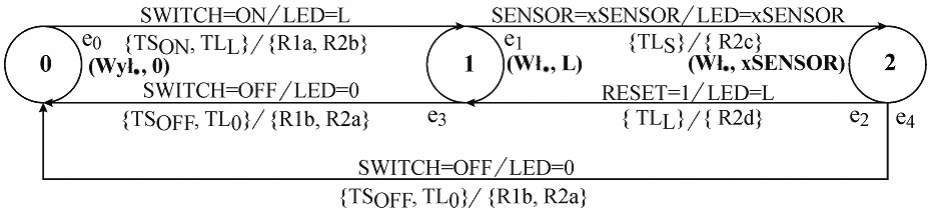
CId	Typ	($t_{min.}$, t_{max})	TId	Warunki
C1	P	(0, 2s)	–	{@T(SWITCH = ON) ∨ @T(RESET = 1)} → {@T(LCD = L)}
C2	Z	(0, 15s)	TL_S	–
C3	Z	(0, 2s)	TL_0, TS_{OFF}	{@T(SWITCH = ON)} →

Ponieważ działanie systemu polega na wykonywaniu zadań, zatem założono, że błąd projektowy powoduje nieprawidłowe wykonanie zadania [8], a jego rezultatem jest inny niż oczekiwany wynik zadania (błąd funkcjonalny) i/lub nieosiągnięcie minimalnego lub przekroczenie maksymalnego czasu przeznaczonego na wykonanie podzbioru zadań (błąd czasowy).

2.2. Graf wymagań funkcjonalnych

Proces generacji scenariuszy testowych korzysta z odrębnego modelu systemu (niezależnego od modeli projektowych). Jest on generowany automatycznie na podstawie formalnej specyfikacji wymagań funkcjonalnych uzupełnionej danymi o zadaniach. **Graf wymagań funkcjonanych** (GWF) [18] bazuje na idei automatu (popularnej wśród metod generacji testów, np. [6, 11, 12]), w którym przepływ danych jest modelowany poprzez zadania. Relacje czasowe nie są w nim bezpośrednio modelowane.

Na rycinie 2 pokazany jest model GWF utworzony dla systemu CO2 na podstawie tabl. 1 i 2. Węzły GWF reprezentują stany systemu i są etykietowane ciągami wartości zmiennych. Węzeł 0 reprezentuje stan początkowy systemu (SYSTEM = Wył., LCD = 0). Etykiety krawędzi grafu GWF, reprezentujących przejścia pomiędzy stanami, zawierają zdarzenia (inicjujące/końcowe – nad strzałkami) oraz identyfikatory wymagań funkcjonalnych (RId) i realizujących je zadań (TId) (pod strzałkami).



Ryc. 2. Graf GWF dla systemu CO2

Fig. 2. GWF graph for CO2 system

Idea prezentowanego rozwiązania problemu generacji scenariuszy testowych opiera się na koncepcji **ścieżki krytycznej** [17] wiążącej ograniczenia czasowe ze ścieżkami w grafie GWF oraz na pomyśle reprezetowania wymagań funkcjonalych jako krawędzi w modelu grafowym [6] zaadaptowanym w formie **ścieżki funkcjonanlej**.

Zdarzenia etykietujące ścieżkę funkcjonalną SF_{RId} inicują wykonanie związanych z nią zadań realizujących wymaganie RId, zatem scenariusz testowy zawierający te zdarzenia umożliwa sprawdzenie poprawności wykonania tych zadań pod względem funkcjonalnym (czyli poprawności wymagania RId). Podobny związek można zaobserwować w przypadku ograniczeń czasowych i ścieżek krytycznych. Do zbadania poprawności czasowej wykonania zadań etykietujących ścieżkę krytyczną SK_{CId} (czyli poprawności ograniczenia CId) dodatkowo wymagana jest znajomość czasu pojawienia się pierwszego zdarzenia inicjującego i ostatniego końcowego sekwencji zdarzeń etykietujących tę ścieżkę.

Powyższe zależności prowadzą do następujących dwóch wniosków:

- walidcja ścieżki krytycznej SK_{CId} jest równoznaczna walidacji ograniczenia czasowego CId,
- walidcja ścieżki funckjonalnej SF_{RId} jest równoznaczna walidacji wymagania funkcjonalnego RId.

Wyznaczenie kompletnego (tj. zapewniającego możliwość walidacji każdego wymagania funkcjonlanego i czasowego) i zredukowanego zbioru spójnych (tj. gwarantujących ciągłość działania systemu dla kolejno pojawiających się zdarzeń) scenariuszy testowych

polega na połaczeniu ze sobą odpowiednich ścieżek funkcjonalych i krytycznych wyznaczonych dla poszczególnych wymagań funkcjonalnych i czasowych. Wyboru ścieżek dokonuje się na podstawie badania ich **pokrycia** (tj. porównania podzbiorów zadań reprezentowanych przez ścieżki związane z jednym wymaganiem funkcjonalnym lub czasowym pod względem wzajemnego zawierania się tych podzbiorów [18]), a efektem końcowym tej selekcji jest semiminimalny zbiór ścieżek – zbiór, który zawiera co najmniej po jednej ścieżce funkcjonlanej dla każdego wymagania funkcjonlanego i wszystkie niepokryte ścieżki dla każdego ograniczenia czasowego.

2.3. Drzewo scenariuszy testowych

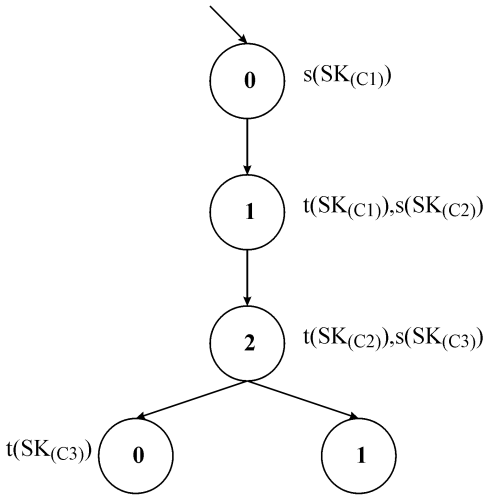
Uzyskanie zbioru spójnych scenariuszy o minimalnym rozmiarze wymagałoby rozpatrzenia wszystkich kombinacji połaczeń ścieżek dla wszystkich semiminimalnych podzbiorów, co jest raczej nieosiągalne w praktyce. Niemniej możliwe jest wyznaczenie zbioru zredukowanego (przybliżenie minimum) przez dynamiczny wybór ścieżek, tj. o wyborze i generacji określonej ścieżki krytycznej lub funkcjonlanej decydują aktualne uwarunkowania, czyli możliwość połączenia ścieżki ze ścieżkami wyznaczonymi wcześniej oraz zapotrzebowanie na daną ścieżkę (jeśli jest niezbędna ze względu na kompletność podzbioru). Podejmowanie decyzji o wyborze i wygenerowaniu ścieżki krytycznej lub funkcjonalnej wspomaga procedura tworzenia **drzewa scenariuszy testowych** (DST). DST jest rozwinięciem grafu GWF w drzewo, którego poszczególne gałęzie opisują różne zachowania systemu. GWF jest rozwijany, dopóki DST nie uwzględnia wszystkich istotnie różnych (bez powtórzeń) zachowań systemu.

Na rycinie 3 pokazane jest drzewo DST uzyskane dla systemu CO2 na podstawie ryc. 2. Węzły, obok których znajdują się oznaczenia w postaci $s(SK_{(CId)})$ i $t(SK_{(CId)})$, to odpowienio węzły rozpoczynające i kończące ścieżki krytyczne wyznaczone dla poszczególnych ograniczeń czasowych CId. Stan 0 jest wyróżnionym stanem początkowym.

Decyzja dotycząca wyboru ścieżki do wyznaczenia jest podejmowana na podstawie aktualnego stanu drzewa DST (tj. czy rozważany węzeł DST rozpoczyna dowolną ścieżkę związaną z niezbadanym jeszcze wymaganiem), a jej konsekwencją zwykle jest eliminacja innych ścieżek, które z chwilą dodania tej wybranej stają się zbędne. Efektem selekcji ścieżek dokonanej w trakcie tworzenia DST i generacji ścieżek jest redukcja ich liczby (w odniesieniu do liczby wszystkich możliwych ścieżek), co skutkuje redukcją uzyskanego zbioru scenariuszy testowych.

Sekwencja zdarzeń uzyskana dla pojedynczej gałęzi DST tworzy jeden **funkcjonalny scenariusz testowy** (FST), a dane czasowe – odpowiadającą mu **ogólną ramówkę czasową** (ORC).

W tablicy 4 pokazano przykładowy scenariusz testowy uzyskany dla systemu CO2. W pierwszej kolumnie (FST1) znajduje się sekwencja zdarzeń (inicjujące/końcowe), a w drugiej (ORC1) symbolicznie zapisane momenty czasowe (t_i/t_j), w których pojawiają się kolejne zdarzenia oraz informacje określające sposób wyliczenia czasu wykonywania zadań dla poszczególnych ograniczeń czasowych i sprawdzenia poprawności obliczonego czasu (obliczenia mogą być wykonane dopiero w trakcie symulacji, po uzyskaniu rzeczywistych wartości czasu). Podczas generacji scenariuszy wartość symboliczna wejścia SENSOR została zastąpiona konkretną wartością wybraną przez projektanta, a wartość na wyjściu LCD została wyliczona na podstawie równania charakteryzującego zadanie TL_S.



Ryc. 3. Drzewo DST dla systemu CO2

Fig. 3. DST tree for CO2 system

Tablica 4

Scenariusz testowy ST1 dla systemu CO2

ST1		
FST1	ORC1	
	t_j/t_i	[CId]: $t_j - t_i \in (t_{min}, t_{max})$
SWITCH = ON/LCD = L SENSOR = 1500 = /LCD = 1500 SWITCH = OFF/LCD = 0	$t1/t2$ $t3/t4$ $t5/t6$	[C1]: $t2 - t1 \in$ (0,2 s) [C2]: $t4 - t3 \in$ (0,15 s) [C3]: $t6 - t5 \in$ (0,2 s)

Scenariusz ST1 pokrywa gałąź 0 → 1 → 2 → 0 w DST. Cała procedura wyboru i generacji ścieżek oraz zapisu scenariuszy jest wykonywana automatycznie z zastosowaniem trzech algorytmów podanych w [S07].

3. Motywacja

Zarysowana w rozdz. 2 metoda generacji zredukowanego zbioru scenariuszy testowych bazuje na dobrze znanych technikach analizy grafów i drzew i daje dobre rezultaty [19]. Jednak ręczne wykonanie wszystkich obliczeń, choć teoretycznie możliwe, nawet dla małych systemów jest bardzo trudne do zrealizowania. Każdorazowe podjęcie decyzji o wyborze ścieżki do wyznaczenia, generacja ścieżek czy badanie ich pokrycia wiąże się z koniecznością wielokrotnego sprawdzenia różnych warunków i powtarzania tych samych czynności, co jest zajęciem bardzo żmudnym, pracochłonnym i długotrwałym, a dodatkowo również bardzo podatnym na błędy. Rozwiązaniem tego problemu jest automatyzacja

metody zwalniająca projektanta z konieczności wykonywania tych wszystkich czynności i znacznie przyśpieszająca cały proces, a ponadto sprawiająca, że scenariusze są wyznaczane w systematyczny, zdyscyplinowany i obiektywny sposób ograniczający wpływ „czynnika" ludzkiego (głównego źródła błędów). Z drugiej strony złożoność obliczeniowa opracowanych algorytmów jest duża – rzędu $O(|C|*|T|*|V|^9)$ [S07], a to wzbudza wątpliwość co do możliwości otrzymania wynikowego zbioru scenariuszy testowych w rozsądnym czasie dla systemów spotykanych w praktyce. Jednak złożoność szacowno dla przypadku najgorszego, tj. przy założeniu najwyższego możliwego kosztu wykonania każdej operacji. Wystąpienie takiej sytuacji w odniesieniu do rzeczywistych systemów jest mało prawdopodobne. Sprawne narzędzie programistyczne implementujące opracowaną metodę może być zatem bardzo pomocne w praktyce, jak również może dostarczyć informacji niezbędnych do oceny praktycznej przydatności opracowanej metody.

4. Kreator ST

Kreator ST [14] jest prototypowym narzędziem implementującym opracowaną metodę wyznaczania scenariuszy testowych. Podstawowe funkcje tej aplikacji pokrywają się z głównymi etapami procesu generacji.

Kreator ST wspomaga wykonanie następujących zadań:

- wprowadzenie, edycja i modyfikacja specyfikacji wymagań systemowych,
- generacja modelu GWF,
- tworzenie drzewa DST,
- zapis zredukowanego zbioru scenariuszy testowych.

Pracę z aplikacją rozpoczyna się od wprowadzenia danych opisujących system – portów, zmiennych, zadań i ich charakterystyk oraz formalnej specyfikacji wymagań funkcjonalnych i ograniczeń czasowych w rozszerzonej notacji SCR. Kreator ST udostępnia proste edytory graficzne dostosowane formatem do każdego rodzaju podawanych informacji, np. tabele SCR, edytor równań dla charakteryzowania zadań itp.

W aktualnej wersji aplikacji niektóre z konstrukcji stosowanej notacji SCR są akceptowane jedynie w ograniczonym zakresie (np. definiowane wartości muszą być typu liczbowego lub wyliczeniowego, dopuszczone są tylko proste warunki, które nie zawierają wyrażeń arytmetycznych i odwołań do wartości symbolicznych, nie istnieje możliwość zdefiniowania warunków pomocniczych dla ograniczeń typu Z) lub są uproszczone (np. bezpośrednie powiązanie równań charakteryzujących zadania ze zdarzeniami). Niemniej w trakcie opracowywania specyfikacji zbadanych dotychczas systemów problemy wywołane tymi ograniczeniami udawało się rozwiązać.

Utworzenie kompletnej specyfikacji systemowej jest obecnie najbardziej pracochłonnym i czasochłonnym etapem pracy z Kreatorem ST. Jest to też jedyny etap, na którym aplikacja może tylko wspomagać (udostępniając odpowiednie edytory) czynności wykonywane ręcznie przez użytkownika. W kolejnych etapach udział użytkownika jest już niewielki i sprowadza się głównie do zainicjowania zadań realizowanych w ich obrębie.

Na podstawie specyfikacji portów, zmiennych i wymagań funkcjonalnych (wraz z identyfikatorami wymagań i zadań), po wyborze odpowiedniej opcji z menu, Kreator ST automatycznie generuje model GWF i udostępnia go w postaci tekstowej (tabele opisujące węzły i krawędzie) oraz w formie graficznej.

Kolejnym etapem pracy z Kreatorem ST jest budowa drzewa DST oraz wyznaczanie ścieżek krytycznych i funkcjonalnych. Poszczególne czynności związane z wyborem, generacją i łączeniem odpowiednich ścieżek są wykonywane automatycznie według opracowanych algorytmów. W tym kroku użytkownik jest zobowiązany do dostarczenia konkretnych wartości dla portów, dla których zdefiniowane zostały wartości symboliczne. Konkretne wartości dla zmiennych i wyjść są przeliczane automatycznie na podstawie równań charakteryzujących zadania. Niestety, Kreator ST nie umożliwia powiązania różnych wartości z poszczególnymi ograniczeniami czasowymi, co dawałoby użytkownikowi większą kontrolę nad generacją DST.

Po utworzeniu drzewa DST Kreator ST automatycznie przechodzi do ostatniego etapu, którym jest zapisanie scenariuszy testowych. Wyniki dwóch ostatnich etapów są dostępne w postaci tekstowej (scenariusze testowe) oraz graficznej (drzewo DST).

Kreator ST, w trakcie wykonywania poszczególnych zadań, generuje obszerne raporty zawierające informacje na temat przebiegu generacji modelu i drzewa DST, istniejących i eliminowanych ścieżek oraz – szczególnie ważne – czasu wykonania poszczególnych obliczeń. Na podstawie tych danych możliwe jest dokładne przeanalizowanie wyników i sposobu ich uzyskania.

5. Badania eksperymentalne

O ile w [19] zbadano jakość generowanych scenariuszy testowych, to celem tych badań było zebranie informacji o przebiegu generacji scenariuszy testowych, tak aby była możliwa ocena aspektów praktycznych samej metody oraz jej implementacji w postaci Kreatora ST. W eksperymencie wykorzystano 6 systemów – dwie centrale monitoringu domowego (CMD1 i CMD2), automat do napojów (DA2), system regulacji oświetlenia (OS), system regulacji ciśnienia w reaktorze (SIS1) oraz system pomiaru stężenia CO_2 (CO2). Główne parametry charakteryzujące badane systemy zebrano w pierwszej części tabl. 5 – są to liczba wejść (We) i wyjść (Wy), liczba wymagań funkcjonalnych (|R|) i ograniczeń czasowych nałożonych na system (|C|) oraz liczba zadań i ich trybów uruchomieniowych (|T|/). W drugiej części tabl. 5 zamieszczono dane dotyczące wygenerowanych modeli GWF, tj. liczbę węzłów (|V|) i krawędzi (|E|) oraz liczbę zmiennych stanu z zaznaczeniem liczby zmiennych, dla których zdefiniowano wartości symboliczne (|Var|/).

Tablica 5

Charakterystyka systemów i ich modeli GWF

Nazwa	Charakterystyka systemu					Charakterystyka GWF		
	We	Wy	|R|	|C|	|T|/	|V|	|E|	|Var|/
CMD1	4	4	22	8	6/12	15	32	6/1
CMD2	5	4	24	9	6/12	15	41	6/1
DA2	3	3	18	4	6/10	5	10	7/4
OS	3	2	13	4	4/7	4	10	4/1
SIS1	4	1	15	5	4/7	5	12	4/0
CO2	3	1	6	3	2/5	3	5	2/1

W tablicy 6 zestawiono wyniki uzyskane w trakcie generacji drzewa DST oraz dane dotyczące wyznaczonego, zredukowanego zbioru scenariuszy testowych. W pierwszej części tabl. 6 podano liczbę istniejących (|SK|) i wyznaczonych (|SK|/) ścieżek krytycznych (dla tych ostatnich podano także ich sumaryczną długość), liczbę ścieżek funkcjonalnych nie zawartych w ścieżkach krytycznych i liczbę pokrytych przez nie wymagań funkcjonalnych (|SF|/|R|) oraz rozmiar ($|DST|_E$ – liczba krawędzi) i wielkość ($|DST|_G$ – sumaryczna długość wszystkich gałęzi) utworzonego drzewa DST. W dugiej części tabl. 6 znajduje się liczba scenariuszy w tym zbiorze (|ST|) oraz rozmiar zbioru ($|S_R|$). Dodatkowo, dla porównania w ostatniej kolumnie podano rozmiar zbioru wyczerpującego (tj. uzyskanego z pominięciem selekcji ścieżek).

Tablica 6

Podsumowanie wyników

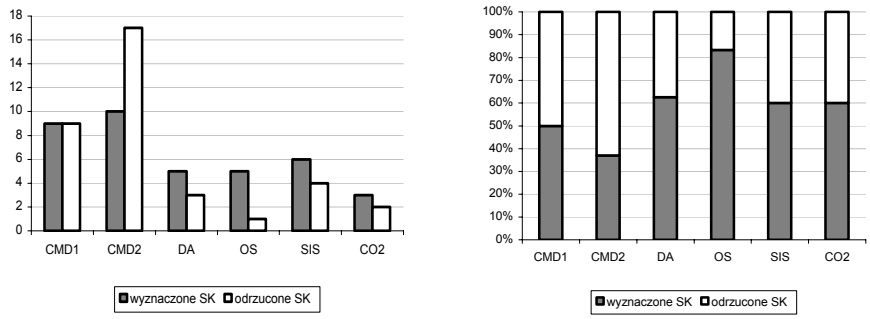
Nazwa	Generacja DST					Scenariusze testowe		
	|SK|	$\|SK\|_G$/	|SF|/|R|	$\|DST\|_E$	$\|DST\|_G$	|ST|	$\|S_R\|$	$\|S_E\|$
CMD1	18	9/12	3/4	29	43	7	43	102
CMD2	27	10/13	3/4	26	35	6	35	163
DA2	8	5/5	1/1	7	9	3	9	13
OS	6	5/15	1/3	16	20	4	20	170
SIS1	10	6/8	0/0	8	13	4	13	26
CO2	5	3/3	1/1	4	6	2	6	8

Ważnych informacji, z punktu widzenia praktycznej użyteczności kreatora, dostarczają dane o liczbie ścieżek krytycznych. Główną przyczyną dużej czasochłonności wykonania opracowanych algorytmów może być (według oszacowań dla najgorszego przypadku) duża liczba istniejących ścieżek krytycznych oraz kosztowne badanie ich pokrycia. Jak pokazano w tabl. 6, rzeczywista liczba ścieżek krytycznych jest stosunkowo mała, co pozwala przypuszczać, że dla rzeczywistych systemów opracowana metoda umożliwi otrzymanie scenariuszy testowych w zadowalającym czasie.

Na podstawie uzyskanych danych zaobserwowano także, że w przypadku każdego systemu, w trakcie generacji drzewa DST można było odrzucić pewną liczbę ścieżek krytycznych. Eliminacja ścieżek jest głównym czynnikiem umożliwiającym redukcję rozmiaru zbioru scenariuszy testowych, zatem jest to ważna obserwacja. Na rycinie 4 pokazano, liczbowo i procentowo, relację pomiędzy liczbą wyznaczonych i odrzuconych ścieżek krytycznych.

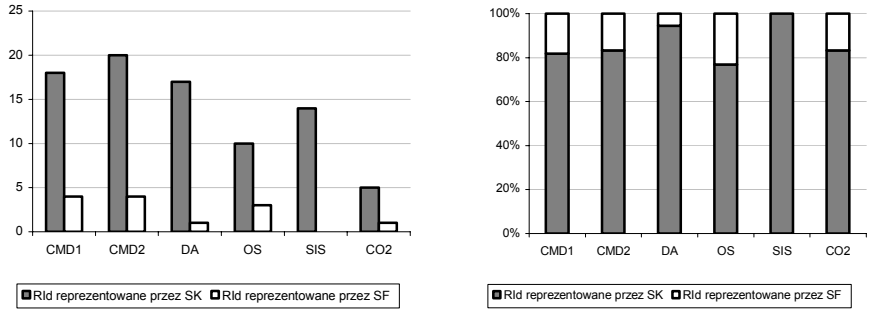
Średnio bezwzględna liczba odrzuconych ścieżek nie jest bardzo duża, choć np. dla systemu CMD2 przewyższyła liczbę ścieżek wyznaczonych. Niemniej jest to dobry rezultat, biorąc pod uwagę fakt, że odrzucenie każdej ścieżki oznacza redukcję końcowego zbioru scenariuszy o wartość odpowiadającą długości tej ścieżki i długości sekwencji łączącej jej węzeł źródłowy z korzeniem DST.

Ścieżki krytyczne wyznaczane w trakcie tworzenia drzewa DST zawsze zawierają pewną liczbę ścieżek funkcjonalnych i z tego powodu przyjęto zasadę pierwszeństwa dodawania ścieżek krytycznych (celem było ograniczenie konieczności oddzielnego analizowania wymagań funkcjonalnych). Na rycinie 5 pokazano relację pomiędzy liczbą wymagań funkcjonalnych związanych ze ścieżakami krytycznymi i z oddzielnymi ścieżkami funkcjonalnymi (w tym także ścieżkami funkcjonalnymi dodanymi ze względu na konieczność zachowania ciągłości w DST).



Ryc. 4. Redukcja ścieżek krytycznych

Fig. 4. Reduction of critical paths



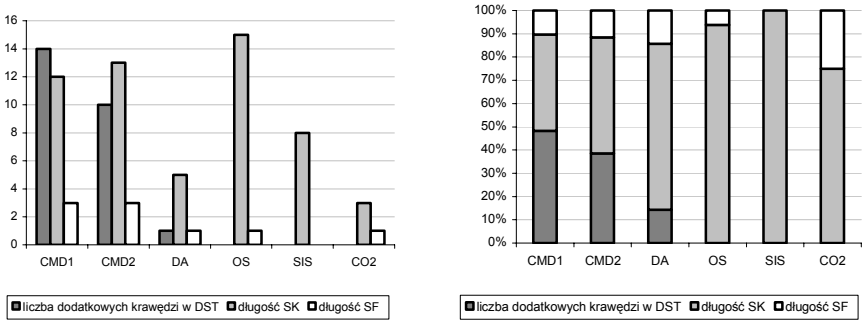
Ryc. 5. Wymagania funkcjonalne reprezentowane przez ścieżki krytyczne i funkcjonalne

Fig. 5. Functional requirements represented by critical and functional paths

W badanych systemach średnio 13,4% wymagań funkcjonalnych nie było reprezentowane przez ścieżki krytyczne, a dla 70% z nich pojawiła się konieczność wyznaczenia odrębnych ścieżek funkcjonalnych. Ścieżki dla pozostałych 30% wymagań zostały dodane do DST podczas dołączania do niego ścieżek krytycznych (jako dodatkowe krawędzie zapewniające ciągłość), więc ich analiza nie wymagała dodatkowego nakładu pracy. Korzystnym zjawiskiem jest także to, że w wielu przypadkach wyznaczenie pojedynczej ścieżki funkcjonalnej umożliwiało reprezentowanie więcej niż jednego wymagania funkcjonalnego. Nawet w przypadku systemu OS wszystkie wymagania funkcjonalne, które nie były związane z żadną ścieżką krytyczną (ponad 20%) były reprezentowane przez jedną ścieżkę funkcjonalną dodaną do DST podczas łączenia ścieżek krytycznych. W najlepszym z zaobserwowanych przypadków, dla systemu SIS, wyznaczone ścieżki krytyczne zawierały ścieżki funkcjonalne reprezentujące wszystkie wymagania funkcjonalne.

Dodawanie do DST krawędzi, które nie należą do żadnej ścieżki krytycznej, a jako ścieżki funkcjonalne są nadmiarowe (tj. istnieje już ścieżka reprezentująca związane z nimi wymagania), ma na celu zapewnienie ciągłości w DST podczas łączenia ścieżek. Powoduje to jednak wzrost rozmiaru zbioru scenariuszy testowych. Na rycinie 6 zilustrowano sposób,

w jaki kształtują się zależności pomiędzy sumaryczną długością wyznaczonych ścieżek funkcjonalnych i krytycznych oraz liczbą takich dodatkowych krawędzi.

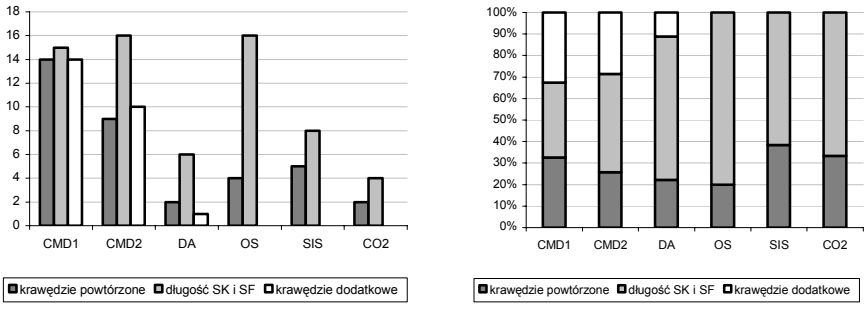


Ryc. 6. Podział DST na ścieżki krytyczne, ścieżki funkcjonalne i krawędzie dodatkowe

Fig. 6. Critical paths, functional paths and the additional edges contained in DST

Sumaryczna długość ścieżek funkcjonalnych dla poszczególnych systemów stanowi średnio 11,4% rozmiaru DST. W większości przypadków ścieżki krytyczne i funkcjonalne obejmują większą część rozmiaru drzewa – średnio 96,5% dla systemów DA, OS, SIS i CO2. Jednak w przypadku systemów CMD1 i CMD2 procentowy udział dodatkowych krawędzi w rozmiarze DST to odpowiednio 48,3% i 38,5%. Niestety, na podstawie przeprowadzonych eksperymentów trudno jednoznacznie stwierdzić, które przypadki mogą okazać się częstsze – decydujące znaczenie będzie miała struktura systemu oraz liczba i charakter zdefiniowanych ograniczeń czasowych.

Pomimo eliminacji pewnej liczby ścieżek krytycznych i funkcjonalnych, która skutkuje średnio 50% redukcją rozmiaru zbioru scenariuszy testowych, problemem nadal pozostaje redundancja. Drugą jej przyczyną, oprócz istnienia dodatkowych krawędzi w drzewie DST, są krawędzie wspólne dla różnych gałęzi DST. Na rycinie 7 pokazano redundancję w zredukowanym już zbiorze scenariuszy testowych obliczoną na podstawie rozmiaru i wielkości drzewa DST (różnica $|DST|_G - |DST|_E$ wyznacza liczbę krawędzi powtórzonych) oraz liczby dodatkowych krawędzi.



Ryc. 7. Redundancja w zbiorze scenariuszy testowych

Fig. 7. Redundancy in the set of test scenarios

Redundancja obliczona w odniesieniu do rozmiaru zbioru scenariuszy testowych (w przybliżeniu wielkości drzewa DST) jest duża. W przypadku systemów CMD1 i CMD2 mniej niż 40% sekwencji tworzących scenariusze testowe w rzeczywistości powoduje wykrycie określonych błędów – pozostałe to składniki sekwencji „odpowiedzialnych" za ustawienie warunków odpowiednich do badania określonych zachowań systemu. Dla pozostałych systemów te proporcje są korzystniejsze – redundancja nie przekracza 40%.

Ważnych informacji z punktu widzenia zastosowania metody i Kreatora ST w praktyce, dostarczają dane czasowe. W tablicy 7 podano czas generacji modelu GWF (t_{GWF}) oraz sumaryczny czas utworzenia drzewa DST i zapisania scenariuszy testowych (t). Czas podany jest w sekundach. Wszystkie dane czasowe zebrano podczas badań z zastosowaniem Kreatora ST, wykonanych na komputerze typu PC (procesor Intel Pentium 4 2,65 GHz, pamięć 512 MB). Zawarte w tabl. 7 wartości to średni czas uzyskany z 10 prób. Dokładność pomiaru wyniosła 0,001 s.

Tablica 7

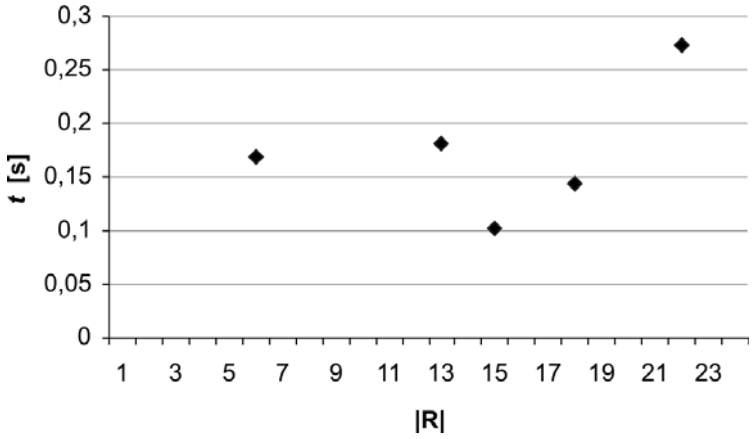
Czas wykonania

Nazwa	t_{GWF}	t
CMD1	0,272	0,61
DA2	0,143	0,20
OS	0,181	0,22
SIS1	0,103	0,15
CO2	0,168	0,4

Na podstawie danych z tabl. 7 opracowano dwa wykresy przedstawiające odpowiednio zależność czasu generacji modelu GWF i scenariuszy testowych od rozmiaru systemu. Na rozmiar systemu wpływa wiele czynników, ale dobrym jego przybliżeniem jest liczba zdefiniowanych wymagań funkcjonalnych i czasowych.

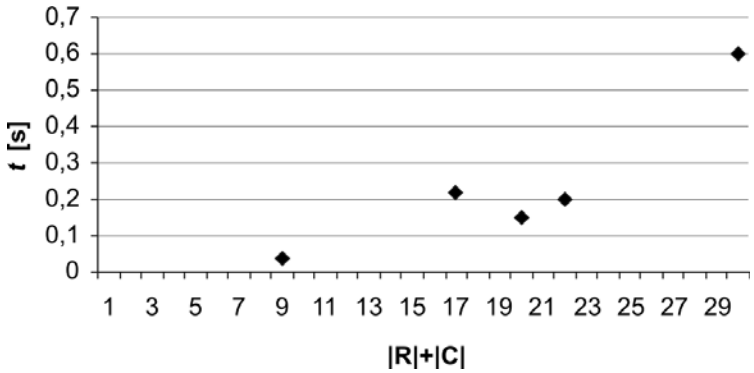
Na rycinie 8 pokazano zależność czasu generacji grafu GWF od liczby wymagań funkcjonalnych (liczba ograniczeń czasowych nie wpływa na model). W niektórych przypadkach czas generacji modelu jest inny niż sugerowałaby to liczba wymagań, np. najlepszy czas zarejestrowano dla systemu SIS (trzeci pod względem liczby wymagań). Do czynników wpływających na czas można zaliczyć liczbę zmiennych oraz liczbę i zakres wartości dla zdefiniowanych wartości symbolicznych. Chociaż na podstawie uzyskanych wyników trudno jest dokładnie określić, jak rozmiar systemu wpływa na czas niezbędny do utworzenia jego modelu, to można sądzić, że w przybliżeniu czas ten rośnie liniowo.

Na rycinie 9 przedstawiono zależność czasu generacji scenariuszy testowych od liczby wymagań funkcjonalnych i ograniczeń czasowych. Na tym wykresie wyraźnie zaznacza się tendencja wzrostowa czasu, chociaż nie jest ona tak gwałtowana jak wynikałoby to z oszacowań złożoności. Niestety, nie sposób jednoznacznie określić jej charakteru, ponieważ na sposób i czas generacji drzewa DST wpływają także indywidualne cechy samych systemów oraz inne czynniki (np. liczba zmiennych i ich wartości, wartości symboliczne i sposób ustalania ich wartości konkretnych itp.), których precyzyjne ujęcie w kategoriach liczbowych nie jest możliwe. Dla systemów DA, SIS i OS, zbliżonych pod względem liczby wymagań i rozmiaru modelu, różnice w czasie generacji scenariuszy nie są duże, choć ewidentnie szybciej uzyskuje się je w przypadku gdy nie zachodzi potrzeba wykonywania dodatkowych obliczeń związanych z ustalaniem konkretnych wartości dla wartości symbolicznych (dla SIS).



Ryc. 8. Zależność czasu generacji GWF od liczby wymagań funkcjonalnych

Fig. 8. GWF generation time related to the number of functional requirements



Ryc. 9. Zależność czasu generacji scenariuszy testowych od liczby wymagań funkcjonalnych i ograniczeń czasowych

Fig. 9. Relation between test scenarios generation time and the number of functional and temporal requirements

6. Podsumowanie

Żadne rozwiązanie jakiegokolwiek problemu z zakresu projektowania systemów nie znajdzie zastosowania w praktyce, jeśli nie będą mu towarzyszyć programistyczne narzędzia wspomagające. Kreator ST jest prototypowym narzędziem implementującym opracowaną metodę generacji scenariuszy testowych. Zastosowanie go podczas eksperymentów wykonanych dla przykładowych systemów znacznie ułatwiło i przyspieszyło uzyskanie konkretnych wyników, a generowane przez Kreator ST raporty umożliwiły szczegółową

analizę całego procesu. Przeprowadzone badania dostarczyły cennych informacji, które pozwoliły na ocenę przyjętych rozwiązań od strony praktycznej. Przede wszystkim pokazano, że złożoność obliczeń dla rozpatrywanych przykładów jest znacznie niższa niż wynikałoby to z oszacowań. Chociaż liczba zbadanych systemów jest zbyt mała, aby można uznać te wyniki za wiążące, to jednak prawidłowości (dotyczące liczby istniejących ścieżek krytycznych) zaobserwowane podczas analizy przebiegu całego procesu dla wszystkich tych systemów (dość zróżnicowanych pod względem indywidulanych charakterystyk) pozwalają sądzić, że taka tendencja może się utrzymywać także dla innych systemów. Uzyskane wyniki potwierdzają również, że eliminacja pewnej liczby ścieżek, na której opiera się redukcja, jest faktem, a nie tylko przypuszczeniem.

Prace nad Kreatorem ST i wykonane eksperymenty pokazały, że zautomatyzowanie całego procesu wyznaczania scenariuszy nie jest zadaniem łatwym. Niektóre etapy generacji scenariuszy okazały się trudne do całkowitego zautomatyzowania, np. wybór konkretnych wartości zastępujących symboliczne wartości przypisane wejściom systemu jest wykonywany przez projektanta. Zaobserwowano także występownie pewnych problemów podczas ustalania i przeliczania konkretnych wartości dla zmiennych. Aktualna wersja Kreatora ST ma ograniczenia, które obecnie nie pozwalają np. na efektywne przedstawienie złożonych wymagań lub powodują niewielkie nieprawidłowości w przebiegu procesu generacji, takie jak np. wielokrotne wyznaczanie i redukowanie takiej samej ścieżki. Niemniej jest to wersja prototypowa i prace nad zwiększeniem jej wydajności są kontynuowane.

Literatura

[1] Alur R., Dill D.I., *A Theory of Timed Automata*, Theoretical Computer Science, Vol. 126, 1994, 183-235.

[2] Cortes L.A., Eles P., Peng Z., *Formal Coverification of Embedded Systems using Model Checking*, Proc. EUROMICRO, 2000.

[3] Clarke E., Emerson E.A, *Synthesis of synchronization skeletons for branching time temporal logic. In Logic of Programs*, Workshop, LNCS, Vol. 131, Springer-Verlag, Yorktown Heights, Nowy Jork, 1981.

[4] Carlson J., *Languages and methods for specifying real-time systems*, MRTC report, Mälardalen Real-Time Research Centre, 2002.

[5] Corno F., Sonza Reorda M., Squillero G., Manzone A., Pincetti A., *Automatic Test Bench Generation for Validation of RT-level Descriptions: an Industrial Experience*, Proc. of DATE, Paryż, Francja, 2000, 385-389.

[6] Cunning S., Rozenblit J.W., *Automating Test Case Generation for Requirements Specification for Real-time Embedded Systems*, Proc. of the 1999 IEEE SMC'99, Tokio, Japonia, 1999, 784-789.

[7] Dasdan A., Ramanathan D., Gupta R.K., *Rate Derivation and Its Applications to Reactive, Real-time Embedded Systems*, Proc. of the 35th Design Automation Conference, San Francisco, USA, 1998, 263-268.

[8] Harris I.G., *Hardware-software covalidation: Fault models and test generation*, IEEE Design and Test of Computers, Vol. 20, 2003, 40-47.

[9] Heitmeyer C., Kirby J., Labaw B., *The SCR Method for Formally Specifying, Verifying and Validating Requirements: Tool Support*, Proc. of the 19th International Conference on Software Engineering, Boston, USA, 1997, 610-611.
[10] Hessel A., Larsen K.G., Nielsen B., Pettersson P., Skou A., *Time--optimal Real-Time Test Case Generation using Uppaal*, proc. of the 3rd International Workshop on Formal Approaches to Testing of Software 2003, LNCS 2931, Springer-Verlag, 2004, 136-151.
[11] Lajolo M., Lavagno L., Rebaudengo M., Sonza-Reorda M., Violante M., *Automatic Test Bench Generation for Simulation-based Validation*, Proc. of the 8th International Workshop on Hardware/Software Co-Design, 2000, 136.
[12] Larsen K.G, Mikucionis M., Nielsen B., Skou A., *Testing real-time embedded software using UPPAAL-TRON: an industrial case study*, in proc. of the 5th ACM International Conference on Embedded Software, Jersey City, USA, 2005, 299-306.
[13] McMillan K.L., *Symbolic Model Checking: An Approach to the State Explosion Problem*, Kliwer 1993.
[14] Maksym P., *Automatyczny weryfikator specyfikacji dla systemów osadzonych*, praca magisterska, Politechnika Krakowska, 2006.
[15] Mishra P., Dutt N., *Automatic Functional Test Program Generation for Pipelined Processors using Model Checking*, Proc. of the 7th IEEE International High--Level Design Validation and Test Workshop, 2002, 99-103.
[16] Strug J., *Automatyczna generacja scenariuszy testowych do walidacji reaktywnych systemów zamkniętych*, rozprawa doktorska, Politechnika Warszawska, 2007.
[17] Strug J., Deniziak S., Sapiecha K., *Zastosowanie scenariuszy testowych do weryfikacji ograniczeń czasowych w systemach zamkniętych*, materiały konferencyjne VI Krajowej Konferencji: „Reprogramowalne Układy Cyfrowe, RUC'2003", Szczecin 2003, 253-25.
[18] Strug J., Deniziak S., Sapiecha K., *Validation of Reactive Embedded Systems against Temporal Requirements*, Proc. of the 18th IEEE ECBS, Brno 2004, 152-159.
[19] Strug J., Sapiecha K., *Wyznaczanie scenariuszy testowych dla reaktywnych systemów wbudowanych*, Systemy czasu rzeczywistego – Kierunki badań i rozwoju, WKiŁ, 2005, 241-250.
[20] Zhang L., Hsiao M., Ghosh I., *Automatic Design Validation Framework for HDL Descriptions via RTL ATPG*, Proc. of the 12th Asian Test Symposium, 2003.