POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko autora rozprawy: Grzegorz Gołaszewski
Dyscyplina naukowa: Informatyka

ROZPRAWA DOKTORSKA

Tytuł rozprawy w języku polskim: Metoda analizy związanych z czasem wymagań dotyczących bezpieczeństwa systemów komputerowych

Tytuł rozprawy w języku angielskim: Safety related timing requirements analysis of computer systems

Promotor *podpis*	Drugi promotor *podpis*
prof. dr hab. inż. Janusz Górski	<Tytuł, stopień, imię i nazwisko>
Promotor pomocniczy *podpis*	Kopromotor *podpis*
<Stopień, imię i nazwisko>	<Tytuł, stopień, imię i nazwisko>

Gdańsk, rok 2017

Zawartość

Spis rysunków

Lista tabel

1 Wprowadzenie

1.1 Bezpieczeństwo systemów komputerowych

Nikogo raczej nie trzeba przekonywać, że bezpieczeństwo jest pożądaną cechą wszystkich systemów. Nie bez powodu producenci samochodów prześcigają się w oferowaniu aktywnych i pasywnych systemów bezpieczeństwa, a nota pięć gwiazdek w testach zderzeniowych Euro NCAP staje się raczej regułą niż szczególnym osiągnięciem [NCAP16]. Jednocześnie w systemach związanych z bezpieczeństwem od dawna już pojawia się oprogramowanie. Można je spotkać w elektrowniach atomowych [PAM91], samolotach [ATSB08], systemach kontroli pociągów [HSC98]. To, że oprogramowanie ma wpływ na niezawodność i bezpieczeństwo systemów, wielokrotnie w historii dowiodły incydenty:

- katastrofy rakiet Ariane 5 [EC96], Cryosat [Bri05],
- awaria sieci telefonicznej firmy AT&T [Bur95],
- awarie maszyny do radioterapii Therac-25 [LT93],
- zapaść londyńskiego systemu ratunkowego [FD96,Dal99].

Dłuższa lista incydentów znajduje się w [Neu86].

Oprogramowanie znajduje jednak zastosowanie w systemach związanych z bezpieczeństwem ze względu na posiadane zalety: uniwersalność, elastyczność, brak ograniczeń fizycznych, możliwość tworzenia dowolnie skomplikowanych reguł sterujących. Te zalety z punktu widzenia funkcjonalnego stają się jednak wadami w ujęciu niezawodnościowym. Np. elastyczność i brak ograniczeń fizycznych oznacza, że wnioskowaniu o działaniu systemu na podstawie ograniczonej liczby próbek towarzyszy większa niepewność niż ma to miejsce w układach fizycznych. Dodatkowo metody wywodzące się z inżynierii oprogramowania i inżynierii bezpieczeństwa były znacząco różne. W dziedzinie analizy bezpieczeństwa oprogramowania poczyniono oczywiście postępy. Nie oznacza to jednak, że badania w tym zakresie staną się kiedykolwiek zbędnie, ze względu choćby na to, że wraz ze wzrostem wydajności systemów komputerowych możliwe staje się tworzenie coraz bardziej skomplikowanego oprogramowania a wraz postępem miniaturyzacji systemy wbudowane stają się coraz bardziej wszechobecne (WSN, Ubiquitous computing).

1.2 Cele i teza rozprawy

Celem rozprawy jest zaproponowanie metody identyfikowania uwarunkowanych czasem wymagań względem systemu komputerowego na podstawie analizy bezpieczeństwa środowiska, z którym system ten współpracuje. Metoda wykorzystuje sformalizowaną technikę analizy drzew niezdatności i wzbogaca ją o technikę analizy scenariuszy hazardów ukierunkowaną na wywiedzenie z tych scenariuszy wymagań wobec systemu komputerowego. Celem jest również opracowanie narzędzi wspomagających zastosowanie proponowanej metody w odniesieniu do rzeczywistych przypadków komputerowych systemów związanych z bezpieczeństwem.

W powiązaniu z przedstawionym powyżej celem postawiono następujące tezy:

1. ***Proponowana metoda umożliwia analizę związanych z czasem scenariuszy hazardów pod kątem identyfikacji uwarunkowanych czasem wymagań bezpieczeństwa wobec systemów komputerowych.***

2. ***Proponowana metoda może być efektywnie zastosowana w odniesieniu do rzeczywistych przypadków komputerowych systemów związanych z bezpieczeństwem.***

Cel został zrealizowany poprzez stworzenie metody TREM (Timing Requirements sElection Method). Metoda ta wykorzystuje:

- analizę drzew niezdatności rozszerzoną o zależności czasowe,
- klasyfikację zdarzeń prostych w drzewie,
- algorytm wywodzenia kandydujących wymagań czasowych.

Celem efektywnego analizowania zależności czasowej w ramach metody TREM zaproponowano cztery algorytmy określania zależności czasowych w Minimalnych Zbiorach Przyczyn (nazywanych również scenariuszami hazardu). Algorytmy te są stosowane zamiennie, różnią się od siebie złożonością obliczeniową, ale charakteryzują również innymi ograniczeniami. Ponieważ w wyniku tej analizy może pojawić się nadmiarowość (np. dwa scenariusze hazardu opisujące ten sam przebieg zdarzeń w czasie), opracowano dwa algorytmy redukcji nadmiarowości w uzyskanych wynikach.

W celu określenia czasowych wymagań wobec systemu zaproponowano sposób klasyfikacji zdarzeń występujących w opisach scenariuszy hazardu oraz algorytm, który na podstawie tych danych przeszukuje scenariusze hazardu celem identyfikacji zależności czasowych, które uniemożliwiłyby wystąpienie danego scenariusza hazardu. W doborze tych zależności uwzględniane są ograniczenia dziedzinowe oraz możliwość wywarcia wpływu na opisywane w takiej zależności zdarzenia poprzez decyzje projektowe. Wszystkie wymienione algorytmy zostały zaimplementowane w ramach PolymorphFT2 – narzędzia wspierającego zaproponowaną metodę.

Celem uzyskania optymalnego algorytmu przetwarzania zależności czasowych konstrukcja algorytmu MZPCALC_FULL została oparta o eksperyment myślowy, w którym porównano możliwe sposoby jego konstrukcji.

Osiągnięte wyniki zostały ocenione w dwóch studiach przypadków, w których ocenione zostały zarówno złożoność obliczeniowa zaproponowanych algorytmów, względna pracochłonność użycia otrzymanych wyników w procesie zapewniania bezpieczeństwa analizowanego systemu oraz potencjał poprawy bezpieczeństwa systemu rozumiany jako odsetek scenariuszy hazardu, dla których udało zidentyfikować wymagania czasowe blokujące ich wystąpienie.

1.3 Struktura rozprawy

Dalsza struktura rozprawy przedstawia się następująco. W rozdziale 2 omówiono zagadnienie bezpieczeństwa systemów komputerowych. Rozdział 3 poświęcony został analizie zależności czasowych w drzewach niezdatności, odniesiono się w nim również do obecnego stanu badań. W rozdziale 4 metoda TREM została ogólnie naszkicowana, po czym wprowadzone zostały wykorzystywane w metodzie modele. W rozdziale 5 rozważone zostały możliwe sposoby analizy

zależności czasowych w drzewach niezdatności. Konkretne algorytmy służące analizie zależności czasowych wprowadzone zostały w rozdziale 6. W rozdziale 7 przedstawiony został sposób wywodzenia wymagań bezpieczeństwa. W rozdziale 8 omówiony został sposób konstrukcji badań walidacyjnych oraz wykorzystane w tym celu narzędzie. Rozdział 9 poświęcony jest weryfikacji proponowanego rozwiązania. Przedstawione są w nim studia przypadków, omawiane są uzyskane wyniki. Rozdział 10 stanowi podsumowanie uzyskanych wyników: wymienia uzyskane rezultaty, omawia osiągnięcie celów, wymienia możliwe dalsze kroki. Rozprawie towarzyszą dwa dodatki: słownik pojęć oraz spis oznaczeń.

2 Bezpieczeństwo systemów

2.1 Podstawowe pojęcia

Bezpieczeństwo jest pożądaną cechą dowolnego obiektu, z którym ludzie mają do czynienia. Niezależnie czy będą to zabawki ([ISO8124, PN71]), pociągi ([PN50128]) czy elektrownie jądrowe ([IAEA]). Intuicyjnie przez bezpieczeństwo obiektu można rozumieć gwarancję nie wyrządzenia krzywdy przez ten obiekt.

Sformułowaną w ten sposób definicję można rozumieć jako absolutną - bezpieczeństwo oznacza stuprocentową gwarancję nie doznania jakiejkolwiek szkody (zdrowotnej, finansowej, czy jakiejkolwiek innej). Tak wyrażone bezpieczeństwo w większości przypadków jest nieosiągalne bądź niepraktyczne. Wystarczy sobie uzmysłowić, że jedynym sposobem uniknięcia szkód w ruchu kołowym jest nie korzystanie z samochodów. Gdyby jednak ograniczyć maksymalną szybkość samochodów do np. 20 km/h, wypadki z udziałem samochodów prawie nigdy nie kończyłyby się śmiercią pasażerów i pojazdy takie zapewne uznawane byłyby za bardzo bezpieczne (jak również za niepraktyczne). Przykład ten umożliwia uzmysłowienie sobie dwóch rzeczy. Po pierwsze, projektując nowe maszyny, konstrukcje czy przedmioty często konieczne jest określenie kompromisu pomiędzy funkcjonalnością a bezpieczeństwem (choć nie jest to reguła). Po drugie bezpieczeństwo nie oznacza wyeliminowania ryzyk, ale ograniczenie ich do akceptowalnego poziomu.

W ten właśnie sposób bezpieczeństwo definiuje norma PN-EN 61508 [PN61508]. Jest to norma określająca ogólne wymagania względem systemów *E/E/PE*[1] związanych z bezpieczeństwem (na podstawie tych wymagań budowane są normy określające wymagania wobec systemów E/E/PE związanych z bezpieczeństwem w konkretnych zastosowaniach {[PN61511, PN61800, PN62061]}).

Norma ta bezpieczeństwo definiuje jako:

> ***bezpieczeństwo (ang. safety) - wolność od nieakceptowalnych ryzyk,***

gdzie ryzyko jest rozumiane jako:

> ***ryzyko (ang. risk) - kombinacja prawdopodobieństwa wystąpienia szkody i ciężkości szkody***
>
> ***szkoda (ang. harm) - obrażenie fizyczne bądź uszczerbek na zdrowiu ludzi bądź zniszczenia mienia bądź w środowisku***

Sposób wyrażania prawdopodobieństwa wystąpienia szkody wydaje się być intuicyjnie zrozumiały. Może to być szansa na wystąpienie szkody przy pojedynczym użyciu bądź szansa wystąpienia szkody w okresie czasu. Ciężkość szkody stanowi większy problem. Dla strat materialnych jej jednostką będzie wybrana waluta. Jednak może to być również liczba zabitych czy zranionych osób. Powoduje to jednak, że różne rodzaje szkód są nieporównywalne. Aby je porównać, trzeba ciężkość szkody wyrazić w tej samej jednostce – na przykład w wybranej walucie. To z kolei skutkuje wystąpieniem

[1] Electric/Electronic/Programmable Electronic (elektryczny/elektroniczny/programowalny)

swego rodzaju konfliktu oraz niejednoznaczności dla wartości uważanych za bezcenne (jak ludzkie życie). W takich przypadkach (śmierć, uszczerbek na zdrowiu, skażenie środowiska itp.) ciężkość szkody może być określana na podstawie konsensusu społecznego (np. ogólnie przyjęte wysokości odszkodowań wypłacanych w takich wypadkach) albo być przedstawiona jako suma wybranych wartości (np. nie więcej niż 1 śmierć/ciężki uszczerbek na zdrowiu rocznie oraz straty materialne nie większe niż 200 000PLN w ciągu roku).

Aby mieć wpływ na ryzyko, konieczne jest określenie źródeł związanych z nim szkód. Norma PN-EN 61508 tych źródeł upatruje w zagrożeniach:

> ***zagrożenie/hazard (ang. hazard) - potencjalne źródło szkody***

Terminem tym objęte są zarówno zdarzenia oddziałujące w krótkim okresie czasu (np. wybuch, pożar), jak i zdarzenia wywierające wpływ na ludzi/mienie/środowisko w sposób ciągły w dłuższym okresie czasu.

Zagrożenie realizuje się w niebezpiecznej sytuacji:

> ***niebezpieczna sytuacja (ang. hazardous situation) - okoliczności w których ludzie, mienie bądź środowisko wystawieni są na jedno bądź więcej zagrożeń***

poprzez niebezpieczne zdarzenie:

> ***niebezpieczne zdarzenie (ang. hazardous event) - zdarzenie które może skutkować szkodą***

Powyższe definicje rozważają bezpieczeństwo w oderwaniu od obiektu, który może być bezpieczny bądź nie. Przedmiotem zainteresowania tej pracy jest bezpieczeństwo systemów. *System* jest definiowany jako zestaw komponentów łącznie z zależnościami pomiędzy nimi, zaprojektowany celem dostarczania określonej usługi [RLT78]. Nie jest wymagane aby komponenty dostarczały usług tylko jednemu systemowi; mogą być składowymi wielu systemów.

System może stać się źródłem zagrożenia w wypadku uszkodzenia. Uszkodzenie można zdefiniować jako ([PN61508]):

> ***uszkodzenie (ang. failure) - ustanie zdolności dostarczania pożądanej funkcji przez jednostkę funkcjonalną bądź działanie jednostki funkcjonalnej w jakikolwiek sposób różny od pożądanego***

Należy zauważyć, że tak rozumiane uszkodzenie jest przejściem od stanu poprawnego dostarczania usługi do niepoprawnego dostarczania usługi [ALRL04]. Przejście w przeciwną stronę można nazwać *przywróceniem usługi*. *Błędem* (ang. error) jest nazywana ta (niepoprawna) część stanu systemu, która może doprowadzić do uszkodzenia. Natomiast ustalona bądź hipotetyczna przyczyna wystąpienia błędu to *niezdatność*. Niezdatność może być *aktywna*, jeśli powoduje błąd w systemie, bądź *uśpiona* (nie powoduje błędu obecnie, ale może do niego doprowadzić w przyszłości).

Norma PN-EN 61508 definiuje niezdatność jako:

> ***niezdatność (ang. fault) - nieprawidłowa okoliczność, która może doprowadzić do redukcji bądź utraty zdolności wykonywania pożądanej funkcji przez jednostkę funkcjonalną***

Powyższa definicja jest różna od przytoczonej poprzednio. Jest to spowodowane faktem, że norma ta pomija pojęcie błędu w opisie powstawania uszkodzenia. Zamiast tego przyjmuje, że uszkodzenie jest powodowane przez niezdatność. Oba źródła zgadzają się odnośnie możliwości istnienia niezdatności bez wystąpienia uszkodzenia (PN-EN 61508 jako przykład podaje błąd w projekcie (ang. design fault)).

Oba modele powstawania uszkodzenia nie są sprzeczne, [ALRL04] podkreśla jednak możliwość upłynięcia dłuższego czasu pomiędzy uszkodzeniem a jego przyczyną (błąd systemu nie objawiający się przez pewien czas w jego zachowaniu obserwowanym z zewnątrz). PN-EN 61508 wprowadza natomiast podział uszkodzeń ze względu na dwa kryteria. Pierwszym kryterium jest natura przyczyny uszkodzenia, ze względu na którą awarię można podzielić na:

> ***losowe uszkodzenie sprzętu[2] (ang. random hardware failure) - występujące w dowolnym momencie czasu uszkodzenie powstałe na skutek jednego lub większej liczby procesów degradacji zachodzących w sprzęcie***
>
> ***uszkodzenie systematyczne (ang. systematic failure) - uszkodzenie powiązane w sposób deterministyczny z pewną przyczyną, które może być wyeliminowane poprzez zmianę projektu, procesu produkcji, procedur operacyjnych, dokumentacji lub innych istotnych czynników***

Należy zauważyć, że zazwyczaj występuje wiele mechanizmów degradacji sprzętu, a ich tempo różni się w poszczególnych komponentach. Skutkiem tego losowe uszkodzenia sprzętu składającego się z wielu komponentów występują w niedających się przewidzieć (losowych) momentach czasu z dającą się przewidzieć częstotliwością. Natomiast częstotliwość uszkodzeń systematycznych trudno jest określić z powodu trudności w przewidzeniu zdarzeń do uszkodzenia prowadzących.

Przyjmując że uszkodzenie jest wynikiem niezdatności, dla obu powyższych rodzajów uszkodzeń można zdefiniować powodujące je niezdatności. Będą to odpowiednio: *niezdatność fizyczna*, czyli degradacja fizycznych komponentów systemu zachodząca w trakcie jego pracy, oraz *niezdatność projektowa* mogąca być określona jako niezdatność systemu obecna przy jego wdrożeniu lub wprowadzona w trakcie aktualizacji, która może skutkować wystąpieniem uszkodzenia.

Powyżej opisano możliwe przyczyny wystąpienia uszkodzenia. Uszkodzenia można jednak klasyfikować ze względu na skutki. Klasyfikacja ta obejmuje dwie pozycje:

[2] random hardware failure można też tłumaczyć jako uszkodzenie sprzętu przypadkowe [PN61511]. Zrezygnowano z tej formy gdyż sugeruje ona niecelowe uszkodzenie sprzętu (tj. przez przypadek) a nie występujące w dowolnym (losowym) momencie.

uszkodzenie niebezpieczne – uszkodzenie systemu bądź jego części, które może skutkować wystąpieniem niebezpiecznego zdarzenia bądź ograniczyć możliwość reakcji systemu na takowe zdarzenie

uszkodzenie bezpieczne – uszkodzenie systemu bądź jego części, które nie ma możliwości przyczynić się do wystąpienia niebezpiecznego zdarzenia oraz do ograniczenia możliwości reakcji systemu na takowe zdarzenie

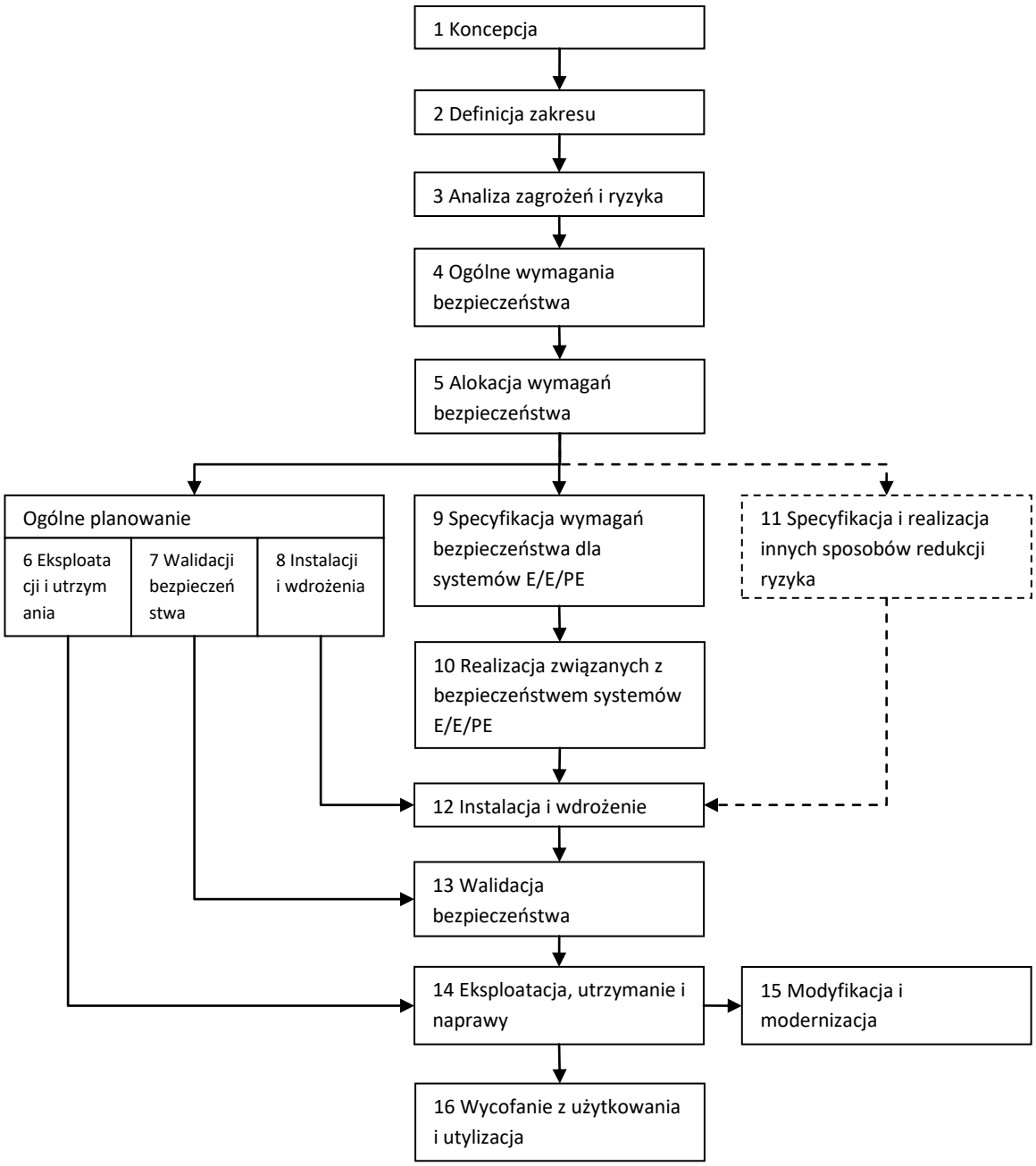
Należy tu nadmienić, że często stosowanym mechanizmem w przypadku (możliwości) wystąpienia uszkodzenia niebezpiecznego jest sprowadzenie systemu do stanu bezpiecznego (stanu o zapewnionym bezpieczeństwie) poprzez wymuszone ograniczenie funkcjonalności systemu – byłoby to uszkodzenie bezpieczne, gdyby nie zostało przeprowadzone celowo. Doskonałym przykładem wymuszania stanu bezpiecznego jest zatrzymanie pociągów przez system „Radio-stop" [IR5]. System ten wymusza zatrzymanie się wszystkich pociągów na określonym obszarze, przez co może pozwolić na uniknięcie kolizji pociągów [Fak14]. Niemniej uruchomienie tego systemu bez wyraźnego powodu jest traktowane jako dywersja i w skrajnym przypadku może doprowadzić do katastrofy [Tvn11] – jest to przykład, że bezpieczeństwo w praktyce nie jest traktowane w sposób bezwzględny.

Jak wspomniano w poprzednim rozdziale podstawą zapewniania bezpieczeństwa systemów w wielu dziedzinach {[PN61511, PN61800, PN62061]} jest norma PN-EN 61508 Bezpieczeństwo funkcjonalne elektrycznych/elektronicznych/programowalnych elektronicznych systemów związanych z bezpieczeństwem [PN61508]. Norma ta opisuje sposób zapewnienia bezpieczeństwa w systemach zawierających układy elektroniczne. Cały proces zapewniania bezpieczeństwa modeluje ona jako *Ogólny cykl życia bezpieczeństwa* (ang. Overall safety lifecycle). Przedstawiony on został na Rysunek 2.1.

Cykl życia bezpieczeństwa przedstawia poszczególne etapy zapewniania bezpieczeństwa w ramach cyklu życia systemu. Dla każdego z tych etapów norma przewiduje cele danego etapu, potrzebne informacje oraz produkty. I tak w etapie koncepcji analizowane jest środowisko tworzonego systemu (fizyczne, prawne, itd.). W etapie drugim określa się zakres tworzonego systemu. Wyniki obu tych etapów wykorzystywane są w kroku trzecim, w którym określa się zagrożenia (zarówno te wynikające z sposobu działania systemu, jak i wynikające ze środowiska, w którym będzie zanurzony). Wskazane jest, że analiza ta ma obejmować wszystkie przewidywalne okoliczności, w tym, co ważne, wszelkie dające się przewidzieć sposoby niewłaściwego wykorzystania (ang. reasonably foreseeable misuse). W tym etapie określane jest również, w jaki sposób może dojść do zagrożenia, oraz z jakim ryzykiem jest związane).

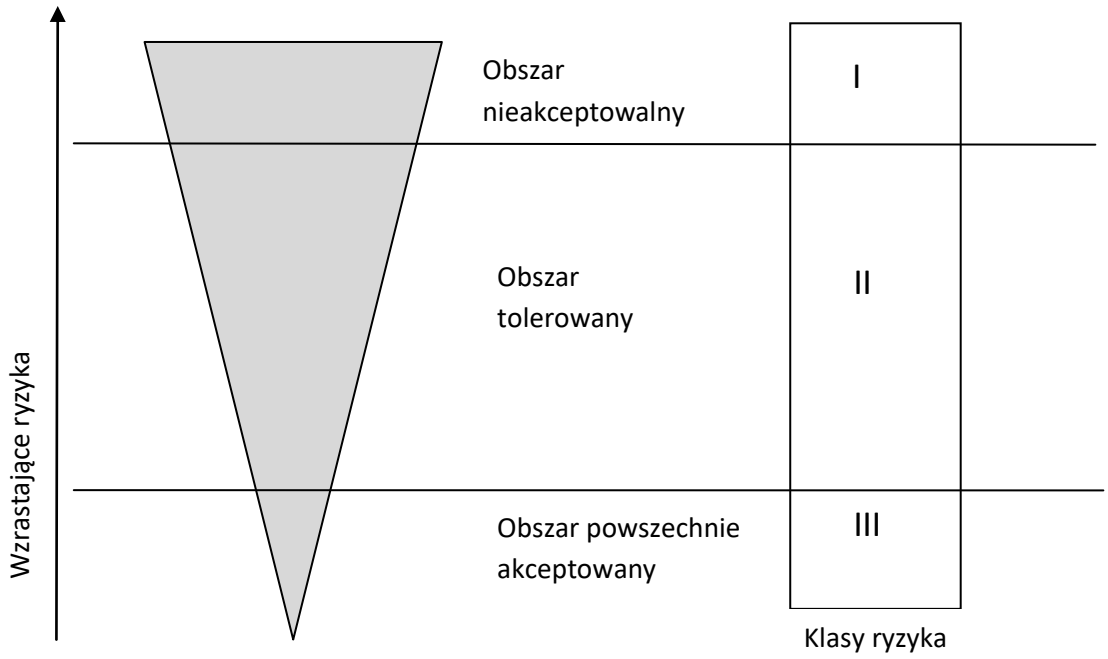
Jak ważne są to etapy najlepiej obrazują wydarzenia mające miejsce 11 Marca 2011r. we wschodniej Japonii. Trzęsienie ziemi o sile 9.0 stopni w skali Richtera i towarzysząca mu fala tsunami zabiło ponad 15000 osób, oraz spowodowało awarię elektrowni atomowej Fukushima Daiichi. W wyniku awarii rozszczelnieniu uległy trzy reaktory, a do środowiska dostało się 6100-12500 PBq substancji radioaktywnych. Raport dyrektora generalnego Międzynarodowej Agencji Energii Atomowej [IAEA15] wskazuje, że przyczyną było niedoszacowanie zagrożeń środowiskowych. Zagrożenie związane z przybrzeżnymi trzęsieniami ziemi oraz towarzyszącymi im falami tsunami było analizowane. Na podstawie lokalnych danych historycznych uznano jednak wystąpienie trzęsienia ziemi o sile 9 stopni

za niemożliwe, choć w innych rejonach o zbliżonej budowie tektonicznej zanotowano nawet silniejsze wstrząsy.



Rysunek 2.1 Cykl życia bezpieczeństwa według PN-EN 61508 [PN61508]

Następny etap (Ogólne wymagania bezpieczeństwa) przekłada wiedzę pozyskaną w poprzednim etapie na wymagania bezpieczeństwa. Na początku poszczególnym zagrożeniom przypisywane jest ryzyko tolerowalne (poziom ryzyka, który może zostać zaakceptowany w danej sytuacji). Ryzyka tolerowalne można wyznaczyć korzystając z metody ALARP (Tak niskie jak praktycznie możliwe). Jest ona powszechnie stosowana przez brytyjską agencję HSE (Health and Safety Executive)[HSE01], w tym dla elektrowni atomowych [HMSO92]. By wykorzystać tą metodę trzeba podzielić ryzyka na 3 klasy przedstawione na Rysunek 2.2.



Rysunek 2.2 Klasy ryzyka i ALARP

Ryzyka z klasy III są w metodzie tej uznawane za szczątkowe i nie wymagające dalszych akcji. Ryzyka z klasy II nazywane są ryzykami niepożądanymi i mogą być tolerowane tylko, jeśli zostaną obniżone tak nisko jak praktycznie możliwe (as low as reasonably practicable). Przy czym im większe ryzyko, tym większe muszą być nakłady w stosunku do osiąganych efektów, aby zrezygnować z obniżania ryzyka. Ryzyka z klasy I nie mogą być zaakceptowane, chyba że zostaną obniżone co najmniej do klasy II.

Jedną z możliwości przypisania ryzyk do kategorii jest zastosowanie tablicy odnoszącej się do prawdopodobieństwa wystąpienia zagrożenia oraz możliwej szkody. Przykład takiej tablicy, zaczerpnięty z [PN61511] przedstawiono w Tabela 2.1.

Tabela 2.1 Przykład klasyfikacji ryzyka

Prawdopodobieństwo	Klasa ryzyka			
	Katastrofalna szkoda	Krytyczna szkoda	Marginalna szkoda	Pomijalna szkoda
Potencjalne	I	I	I	II
Prawdopodobne	I	I	II	II
Możliwe	I	II	II	II
Rzadkie	II	II	II	III
Nieprawdopodobne	II	II	III	III
Niewiarygodne	II	III	III	III

W powyższej tabeli zarówno prawdopodobieństwo ryzyka jak i rozmiar szkody zostały wyrażone słownie. Dla konkretnych zastosowań terminom tym trzeba przypisać wartości liczbowe. Jednocześnie należy zaznaczyć, że tabela ta nie zawsze będzie wyglądać tak samo. Na klasyfikację ryzyk może mieć wpływ rodzaj rozpatrywanego systemu (inny jest poziom akceptacji ryzyk dla elektrowni a inny dla wiertarek). Poziom akceptacji ryzyka zależeć może również od tego, czy narażoną osobą będzie pracownik, czy osoba postronna.

Dodatkowo organizacja wdrażająca system związany z bezpieczeństwem bądź opinia publiczna może kłaść większy nacisk na jeden z aspektów ryzyka kosztem drugiego. W przypadku gdy nacisk kładziony jest na szkodę kosztem prawdopodobieństwa wystąpienia niebezpiecznego zdarzenia mamy do czynienie z ryzykiem uspołecznionym.

> ***ryzyko uspołecznione (ang. societal risk) – ryzyko powstałe gdy na skutek pojedynczego niebezpiecznego zdarzenia może zginąć wiele osób. Nazwa ryzyka związana jest z faktem, że wystąpienie takiego zdarzenia z dużym prawdopodobieństwem wywoła reakcję społeczną oraz polityczną***

Organizacje mogą się cechować awersją do ryzyk uspołecznionych. W takim przypadku klasa ryzyka dla zagrożeń o takim samym poziomie ryzyka, ale różnych wielkościach możliwej szkody będą mogły się różnić.

Znając ryzyka tolerowalne dla poszczególnych zagrożeń można sformułować wymagania bezpieczeństwa. Wymagania te wyrażane są poprzez ogólne funkcje bezpieczeństwa oraz przypisaną im integralność bezpieczeństwa.

> ***funkcja bezpieczeństwa – funkcja mająca być zaimplementowana przez system E/E/PE związany z bezpieczeństwem bądź inny sposób redukcji ryzyka, której celem jest osiągnięcie lub utrzymanie systemu w stanie bezpiecznym, specyfikowana w odniesieniu do konkretnego niebezpiecznego zdarzenia***

Przez *system E/E/PE związany z bezpieczeństwem* rozumiany jest system E/E/PE, który wypełnia funkcje bezpieczeństwa oraz którego działanie ma wpływ na poziom integralności bezpieczeństwa rozpatrywanych funkcji.

Wśród funkcji bezpieczeństwa można wyróżnić ogólne (ogół środków mających na celu osiągnięcie bądź utrzymanie stanu bezpiecznego w odniesieniu do konkretnego ryzyka) oraz elementową (część funkcji bezpieczeństwa implementowana przez poszczególny element systemu).

> ***nienaruszalność bezpieczeństwa(ang. safety integrity) – prawdopodobieństwo, że system satysfakcjonująco wykona wskazaną funkcję bezpieczeństwa we wszystkich wyspecyfikowanych warunkach w zadanym okresie czasu***

Innymi słowy funkcja bezpieczeństwa określa w jaki sposób chcemy zapobiegać wystąpieniu niebezpiecznego zdarzenia i/lub ograniczać powstałą szkodę, a nienaruszalność bezpieczeństwa określa, jak często ta funkcja musi wykonać się poprawnie.

Funkcje bezpieczeństwa projektuje się oraz przypisuje im nienaruszalności bezpieczeństwa tak, aby sprostać wymaganiu:

Wymagany jest taki poziom nienaruszalności bezpieczeństwa systemu E/E/PE związanego z bezpieczeństwem lub innego sposobu redukcji ryzyka, aby:

- średnie prawdopodobieństwo uszkodzenia na wywołanie systemu związanego z bezpieczeństwem było wystarczająco niskie żeby zapobiec przekroczeniu przez częstotliwość występowania niebezpiecznego zdarzenia wartości wymaganej, aby osiągnąć poziom tolerowalnego ryzyka,
- systemy związane z bezpieczeństwem tak modyfikowały konsekwencje uszkodzenia, aby osiągnąć poziom tolerowalnego ryzyka.

Zakłada się, że nienaruszalność bezpieczeństwa składa się z dwóch części:

> ***sprzętowa nienaruszalność bezpieczeństwa (ang. hardware safety integrity) – część nienaruszalności bezpieczeństwa związana z niebezpiecznymi losowymi uszkodzeniami sprzętu***
>
> ***systematyczna nienaruszalność bezpieczeństwa (ang. systematic safety integrity) – część nienaruszalności bezpieczeństwa związana z niebezpiecznymi uszkodzeniami systematycznymi***

Podział ten jest istotny ze względu na to, że techniki zapewniania niezawodności mają różny efekt dla tych dwóch rodzajów nienaruszalności bezpieczeństwa. Nadmiarowość przy wykorzystaniu tych samych elementów dobrze sprawdza się przy zapewnianiu sprzętowej nienaruszalności bezpieczeństwa, nie ma natomiast żadnego efektu w przypadku systematycznej nienaruszalności bezpieczeństwa.

Określenie działań potrzebnych dla zapewnienia żądanej nienaruszalności bezpieczeństwa byłaby trudna przy skali liniowej, dlatego norma PN-EN 61508 [PN61508] dzieli tę skale na dyskretne przedziały nazwane poziomami nienaruszalności bezpieczeństwa.

> ***poziom nienaruszalności bezpieczeństwa, SIL (ang. safety integrity level) – jeden z czterech poziomów odpowiadających określonemu zakresowi wartości nienaruszalności bezpieczeństwa, gdzie poziom 4 oznacza najwyższy poziom nienaruszalności bezpieczeństwa, a poziom 1 najniższy***

Można jeszcze wyróżnić poziom SIL 0, ale poziom ten oznacza brak wymagań wobec nienaruszalności bezpieczeństwa, więc nie jest on „kanoniczny”.

Norma PN-EN 61508 [PN61508] określa przedziały wartości odpowiadające poszczególnym poziomom nienaruszalności bezpieczeństwa, ale aby je podać należy najpierw wprowadzić tryby pracy funkcji bezpieczeństwa. Wyróżnia się trzy takie tryby:

> ***tryb niskiego zapotrzebowania (ang. low demand mode) – tryb pracy funkcji bezpieczeństwa w którym zapotrzebowanie na użycie funkcji jest nie wyższe niż raz na rok***
>
> ***tryb wysokiego zapotrzebowania (ang. high demand mode) – tryb pracy funkcji bezpieczeństwa w którym zapotrzebowanie na użycie funkcji jest wyższe niż raz na rok***

tryb ciągły (ang. continuous mode) – tryb pracy funkcji bezpieczeństwa w którym funkcja jest w ciągłym użyciu

Dla tak zdefiniowanych trybów pracy poziomy nienaruszalności bezpieczeństwa zdefiniowano następująco:

Tabela 2.2 Poziomy nienaruszalności bezpieczeństwa

SIL	Funkcja bezpieczeństwa w trybie niskiego zapotrzebowania	Funkcja bezpieczeństwa w trybie wysokiego bądź ciągłego zapotrzebowania
	Średnie prawdopodobieństwo niebezpiecznego uszkodzenia na wywołanie funkcji bezpieczeństwa (PFD_{avg})	Średnia częstotliwość niebezpiecznego uszkodzenia funkcji bezpieczeństwa $[h^{-1}]$ (PFH)
4	od $\geq 10^{-5}$ do $< 10^{-4}$	od $\geq 10^{-9}$ do $< 10^{-8}$
3	od $\geq 10^{-4}$ do $< 10^{-3}$	od $\geq 10^{-8}$ do $< 10^{-7}$
2	od $\geq 10^{-3}$ do $< 10^{-2}$	od $\geq 10^{-7}$ do $< 10^{-6}$
1	od $\geq 10^{-2}$ do $< 10^{-1}$	od $\geq 10^{-6}$ do $< 10^{-5}$

Zakłada się również, że jeśli dany element systemu implementuje funkcje bezpieczeństwa o różnych poziomach SIL ma być traktowany jako implementujący funkcję o najwyższym poziomie SIL.

Dodatkowo norma PN-EN 61508 [PN61508] nakłada obostrzenia na przypisywanie funkcjom bezpieczeństwa poziomu nienaruszalności bezpieczeństwa SIL 4.

Wymagane jest rozważenie, czy poziomu SIL 4 nie da się uniknąć poprzez:

- wprowadzenie dodatkowych podsystemów związanych z bezpieczeństwem bądź innych form redukcji ryzyka,
- ograniczenie wielkości szkody,
- obniżenie prawdopodobieństwa wystąpienia szkody.

Jeśli natomiast funkcja bezpieczeństwa o poziomie SIL 4 będzie implementowana wymagane jest przeprowadzenie metodami ilościowymi analiza ryzyka obejmująca uszkodzenia o wspólnej przyczynie pomiędzy systemem E/E/PE implementującym funkcje SIL4 a:

- każdym innym systemem, którego uszkodzenie powodowałoby wywołanie analizowanego systemu, oraz
- każdym innym systemem związanym z bezpieczeństwem.

Ponadto zakazane jest przypisywanie funkcjom bezpieczeństwa systemu E/E/PE nienaruszalności bezpieczeństwa wyższej niż w Tabela 2.2 $(10^{-5}/10^{-9}[h^{-1}])$.

W etapie 9 cyklu życia bezpieczeństwa ogólne funkcje bezpieczeństwa dzielone są na funkcje elementowe i przypisywane są (wraz z odpowiednim poziomem SIL) do poszczególnych elementów systemu.

W etapie 10 następuje wytworzenie systemów E/E/PE związanych z bezpieczeństwem na podstawie opracowanych wymagań bezpieczeństwa. Implementacja funkcji bezpieczeństwa nie wymagających użycia sytemów E/E/PE odbywa się w etapie 11 i nie jest regulowana przez omawianą normę.

Dalsze etapy to instalacja, walidacja bezpieczeństwa, użytkowanie i modyfikacja wytworzonego systemu oraz ostatecznie jego wycofanie z eksploatacji.

2.2 Specyfika bezpieczeństwa systemów komputerowych

Zapewnianie bezpieczeństwa systemów komputerowych, a w szczególności oprogramowania jest specyficzne z wielu względów [HSC98]. Po pierwsze, ze względu na ich uniwersalność i brak fizycznych ograniczeń w definiowaniu funkcjonalności, zadania powierzane komputerom mają wyższą złożoność niż funkcje wykonywane przez systemy mechaniczne czy elektryczne. Złożoność ta wyraża się potem w rozmiarze oprogramowania. Francuski system kontroli pociągów SACEM ma około 20,000 linii kodu, Sizewell PPS (podstawowy system ochrony elektrowni atomowej Sizewell) ma 100,000 linii kodu, a typowy przełącznik telefoniczny ponad 10 milionów [HSC98]. Powoduje to oczywiste problemy ze zrozumieniem kodu (typowa powieść ma 10,000 linii).

Ponadto problem stanowi nieciągłe zachowanie dyskretnej logiki, w tym oprogramowania. Dla systemów fizycznych zbliżone warunki powodują zbliżone reakcje. Na przykład, jeśli przy obciążeniu 1kg belka ugnie się o 1 cm, a przy obciążeniu 2kg o 2 cm, to można śmiało założyć, że przy obciążeniu 1,5kg wychylenie wyniesie $(1cm, 2cm)$. Tak samo, jeśli most wytrzyma obciążenie 100 ton, to wytrzyma również 90 ton. Dla programów komputerowych dowolnie mała zmiana parametrów wejściowych może spowodować arbitralnie duże odchylenie wartości wyjściowych. W połączeniu z potencjalnie dużą złożonością, która może uniemożliwić przetestowanie wszystkich możliwych wartości wejściowych stanowi to wyzwanie przy zapewnianiu niezawodności.

Dodatkowym problemem są zmiany. Przewidzenie skutków wprowadzenia zmian w fizycznej strukturze systemu jest w ogólności wykonalne. Zakładać też można, że zmiana będzie rzutowała na elementy bezpośrednio powiązane. Np. jeśli w samochodzie wymienione zostaną sprężyny na dłuższe, to zwiększy się jego prześwit oraz wysokość. Może też zachowywać się gorzej w zakrętach. Nie ma jednak powodu rozważać, czy zmieniła się pojemność bagażnika. W oprogramowaniu nie można a priori zakładać braku wpływu zmian w kodzie na inne elementy systemu. Przykładem może być załamanie długodystansowej sieci telefonicznej firmy AT&T 15 stycznia 1990 roku. Do przywrócenia działania sieci firma AT&T potrzebowała 9 godzin. Przez ten czas nie była w stanie obsłużyć szacowanej liczby 50 milionów połączeń telefonicznych, co kosztowało ją 60 milionów dolarów. Źródło problemu udało się wyśledzić w poprawce oprogramowania wprowadzonej w grudniu w przełącznikach telefonicznych firmy. Zawierała ona błąd powodujący nadpisanie wiadomości w buforze przychodzących wiadomości w sytuacji gdy przełącznik otrzymał dwie wiadomości w krótkim odstępie czasu. Pech chciał, że błąd ujawnił się przy odbieraniu wiadomości administracyjnych informujących o ponownym uruchomieniu innego przełącznika. Błąd ten wychwytywany był przez system korekcji błędów, który wymuszał ponowne uruchomienie przełącznika telefonicznego. W ten sposób w sieci telefonicznej została wywołana reakcja kaskadowa. Poprawka wprowadzona na przełącznikach firmy była przed wprowadzeniem intensywnie testowana, jednak nie przewidziano w jaki sposób może wpłynąć na całą sieć [Bur95].

Kolejną różnicą jest niemożliwość wymiany uszkodzonych elementów. W tradycyjnych systemach wymiana uszkodzonego elementu na nowy identyczny spowoduje, że system będzie znowu funkcjonował poprawnie (np. wymiana pękniętej opony na nową). Wgranie nowej, identycznej kopii oprogramowania w miejsce uszkodzonego nic nie zmieni w działaniu systemu. Nowo wgrane oprogramowanie musi być zmienione celem usunięcia błędu, przez co każde przywrócenie działania oprogramowania jest w istocie zmianą w projekcie systemu i musi zostać jako takie potraktowane. W szczególności sprawdzone, czy nie wprowadza do systemu nowych błędów.

Wszystkie uszkodzenia oprogramowania ze swojej natury są uszkodzeniami systematycznymi, co przekłada się na definicję nienaruszalności bezpieczeństwa oprogramowania normy PN-EN 61508 [PN61508]:

> ***nienaruszalność bezpieczeństwa oprogramowania (ang. software safety integrity) – część nienaruszalności bezpieczeństwa powiązana z systematycznymi uszkodzeniami, których źródłem jest oprogramowanie***

Fakt ten ma dwa ważne następstwa. Pierwszym jest znaczny wzrost trudności w określaniu częstości występowania uszkodzeń. W fizycznych elementach systemu zachodzą procesy degradacji, które są poznane, ale nie można się ich pozbyć. W związku z tym po jakimś czasie każdy element fizyczny ulegnie uszkodzeniu, ale prawdopodobieństwo takiego wydarzenia jest stosunkowo łatwo określalne. W oprogramowaniu każdy poznany błąd najlepiej jest usunąć. W związku z tym określanie częstości uszkodzeń oprogramowania jest de facto określaniem częstości ujawniania się niezdatności nieznanych w momencie analizy niezdatności.

Drugim następstwem jest nieskuteczność stosowania nadmiarowości z użyciem takich samych elementów charakterystyczna dla wszystkich uszkodzeń systematycznych. Dla oprogramowania jednak nawet użycie różnych, stworzonych „niezależnie", wersji oprogramowania nie przynosi tak dobrych efektów, jak przy losowych uszkodzeniach sprzętu.

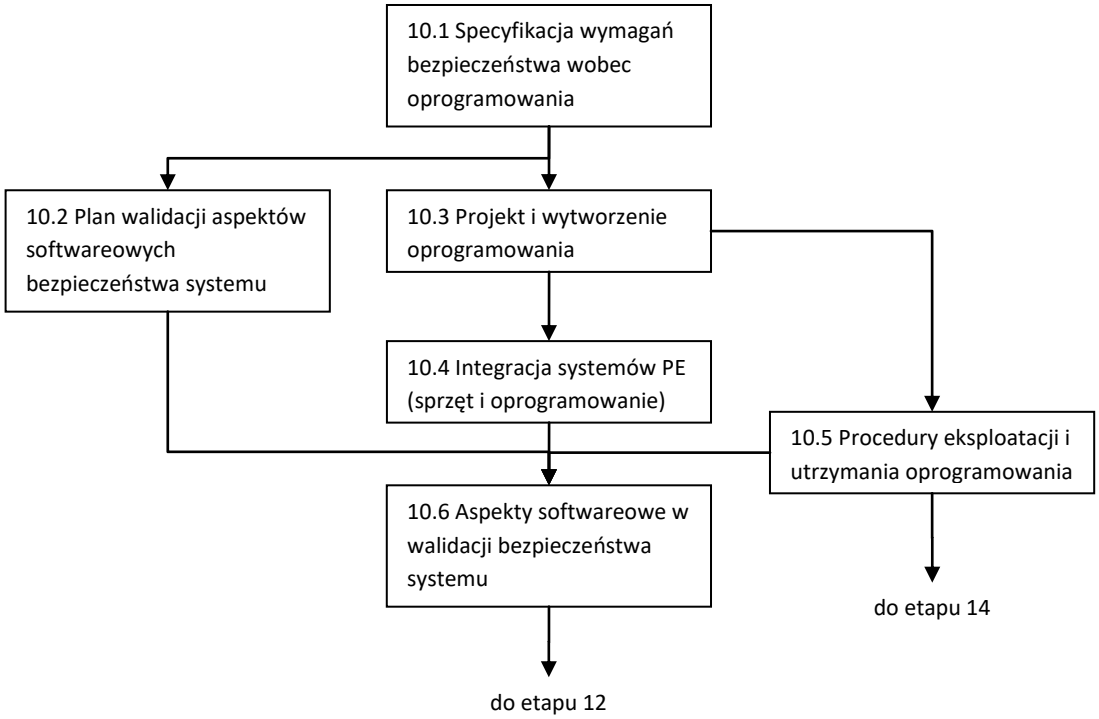
Przy tylu problemach związanych z zastosowaniem oprogramowania w systemach związanych z bezpieczeństwem naturalnym wydaje się dążenie do ponownego wykorzystania sprawdzonych komponentów. Podejście to, choć słuszne, też posiada swoje pułapki. Przy ponownym wykorzystaniu oprogramowania kluczowe jest rozumienie zakresu parametrów dla których oprogramowanie zostało przeznaczone. Trudność w tym aspekcie stanowi wspomniana już wyżej złożoność oprogramowania, szczególnie gdy założenia nie są właściwie udokumentowane. Sztandarowym przykładem skutków zastosowania oprogramowania poza dopuszczalnymi warunkami zastosowań jest wypadek rakiety Ariane 5 ([EC96]). 4 czerwca 1996r. podczas swojego pierwszego lotu rakieta Ariane 5 wybucha 39 sekund po starcie z Kourou w Gujanie Francuskiej. Przyczynę niepowodzenia lotu udało się przypisać do Inercyjnego Systemu Referencyjnego (ang. Inertial Reference System)- SRI. System ten został niemal bez zmian przeniesiony z rakiety Ariane 4. Mechanizm wypadku był następujący: parametr Horizontal Bias (związany z prędkością poziomą) wychodzi poza zakres podczas konwersji z 64 bitowej wartości zmiennoprzecinkowej do 16 bitowej wartości całkowitej ze znakiem. Wyjątek pozostaje nieobsłużony, gdyż w Ariane 4, ze względu na ustaloną trajektorię startową, nie było możliwości by do przepełnienia doszło. Wyjątek powoduje wyłączenie zapasowego, a po chwili aktywnego SRI. To z kolei skutkuje utratą możliwości sterowania lotem i włączenie mechanizmu

autodestrukcji po wejściu na niebezpieczną trajektorię. Aby, jak mówią Anglicy, dodać upokorzenie do obrażeń, okazało się, że szkody wyliczone na 500 milionów dolarów spowodował fragment kodu, który w momencie uszkodzenia nie produkował żadnych przydatnych danych (w Ariane 5 produkowane przezeń dane potrzebne były do momentu startu, tylko w Ariane 4 zachodziła potrzeba ich obliczania jeszcze przez 40 sekund po starcie).

Ze względu na różnice pomiędzy oprogramowaniem oraz układami elektronicznymi zarówno w aspekcie technik wytwarzania, jak i zapewniania bezpieczeństwo, nie będzie zaskoczeniem, że norma PN-EN 61508 [PN61508] dzieli etap 10 - Realizacja związanych z bezpieczeństwem systemów E/E/PE (patrz Rysunek 2.1) na dwa współbieżne strumienie:

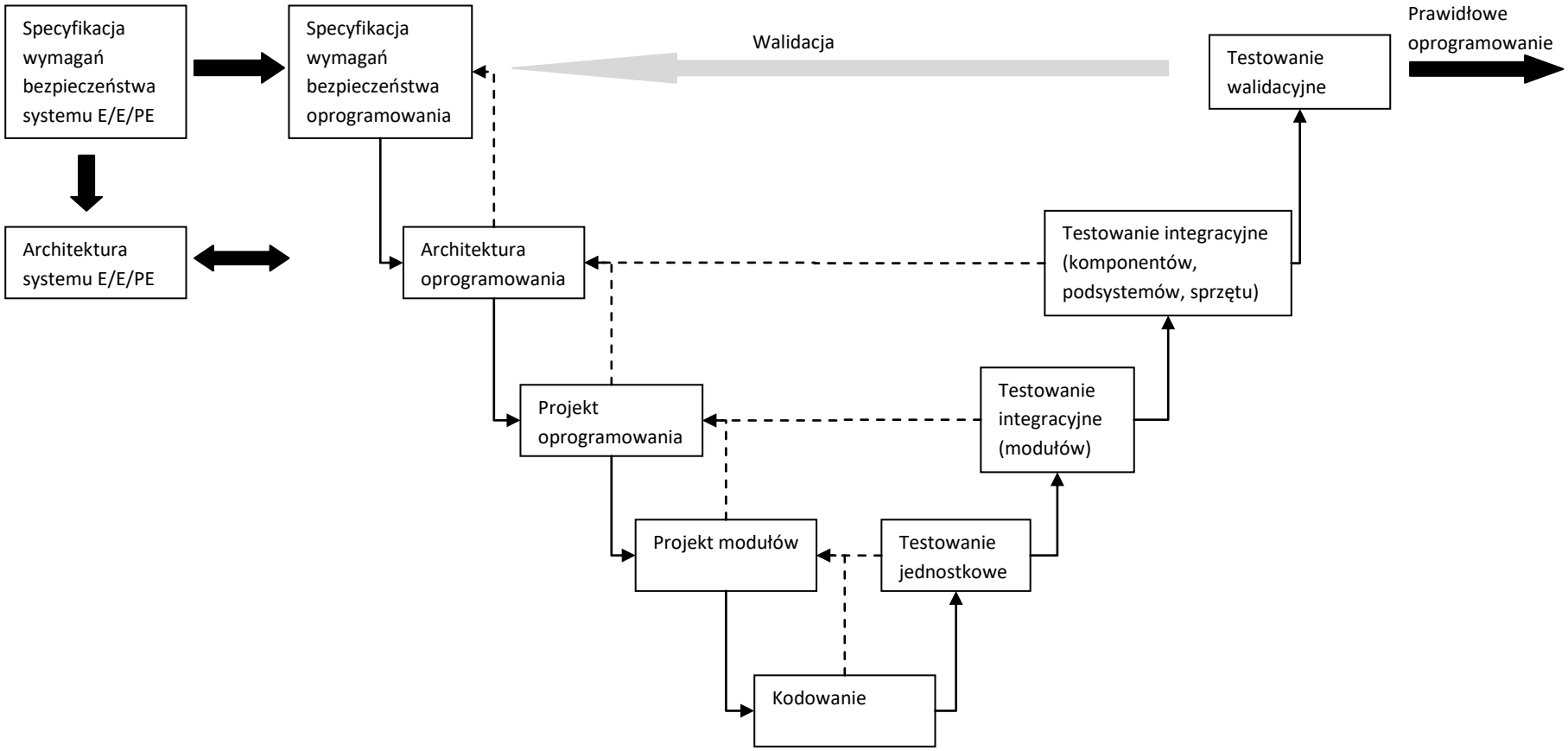
- cykl życia bezpieczeństwa systemów E/E/PE,
- cykl życia bezpieczeństwa oprogramowania.

Rysunek 2.3 przedstawia cykl życia bezpieczeństwa oprogramowania.



Rysunek 2.3 Cykl życia bezpieczeństwa oprogramowania w fazie realizacji według PN-EN 61508 [PN61508]

Cykl życia bezpieczeństwa oprogramowania oraz cykl życia bezpieczeństwa systemów E/E/PE przebiegają równolegle i wzajemnie na siebie oddziałują. Razem tworzą one Etap 10 cyklu życia bezpieczeństwa.



Rysunek 2.4 Wytwarzanie oprogramowania w cyklu życia bezpieczeństwa systemu według PN-EN 61508 [PN61508]

Cele poszczególnych etapów przedstawionych na Rysunek 2.3 są intuicyjnie zrozumiałe i tak etap 10.1 służy specyfikacji wymaganych funkcji bezpieczeństwa oprogramowania (ang. software safety function – funkcja bezpieczeństwa realizowana przez oprogramowanie) oraz powiązanych z nimi poziomów nienaruszalności bezpieczeństwa. Celem etapu 10.2 jest stworzenie planu walidacji tych elementów bezpieczeństwa systemu, które wiążą się z oprogramowaniem. Celem etapu 10.4 integracja oprogramowania z platformą sprzętową, na której ma być osadzone. Celem etapu 10.5 jest dostarczenie niezbędnych informacji i procedur umożliwiających zapewnienie bezpieczeństwa podczas eksploatacji i modyfikacji oprogramowania. Natomiast etap 10.6 służy zapewnieniu, że system spełnia wymagania bezpieczeństwa związane z oprogramowaniem na założonym poziomie nienaruszalności bezpieczeństwa.

W powyższym wyliczeniu celowo pominięto etap 10.3, gdyż dla tego etapu standard PN-EN 61508 dla tego etapu określa cały szereg celów, które (w większości) odnoszą się do kroków modelu V cyklu wytwarzania oprogramowania (patrz Rysunek 2.4):

- architektura oprogramowania: stworzenie architektury oprogramowania , która umożliwi wypełnienie określonych wymagań bezpieczeństwa z uwzględnieniem wymaganego poziomu SIL,
- narzędzia i języki programowania: dobór odpowiednich narzędzi i języków programowania wspomagających pełen cykl życia bezpieczeństwa oprogramowania,
- projekt oprogramowania, projekt modułów, kodowanie: zaprojektowanie i implementacja oprogramowania, które spełnia wyspecyfikowane wymagania bezpieczeństwa z uwzględnieniem wymaganego poziomu SIL i które jest analizowalne, weryfikowalne i może być bezpiecznie modyfikowane,
- testowanie jednostkowe: weryfikacja, że wymagania bezpieczeństwa z uwzględnieniem wymaganego poziomu SIL zostały spełnione; pokazanie, że każdy z modułów oprogramowania wykonuje zamierzone funkcje i nie wykonuje niezamierzonych,
- testowanie integracyjne: weryfikacja, że wymagania bezpieczeństwa z uwzględnieniem wymaganego poziomu SIL zostały spełnione; pokazanie, że wszystkie moduły/komponenty oprogramowania poprawnie współpracują w celu wykonania zamierzonych funkcji i nie wykonują niezamierzonych.

Dla wszystkich omawianych etapów (oraz kroków etapu 10.3) omawiana norma określa zarówno dane wejściowe, produkty, jak i szczegółowe wymagania. Nie będą one jednak omawiane w ramach tej rozprawy.

2.3 Metody analizy bezpieczeństwa

Dla poszczególnych etapów cyklu życia bezpieczeństwa norma PN-EN 61508 przewiduje stosowanie różnorakich metod i technik. Dla zakresu tej pracy szczególnie istotne są techniki analizy uszkodzeń. W kontekście oprogramowania norma PN-EN 61508 zaleca stosowanie takich technik dla poziomów nienaruszalności bezpieczeństwa SIL 1 i SIL 2, a dla poziomów SIL 3 oraz SIL 4 zdecydowanie zaleca (niezastosowaniu wskazanych technik musi towarzyszyć szczegółowe uzasadnienie). Wśród wymienionych technik analizy uszkodzeń wymienione zostały:

- Analiza drzew zdarzeń (ETA, ang. Event Tree Analysis) – indukcyjna metoda służąca określaniu konsekwencji wybranego zdarzenia inicjującego ([Tre16]),
- Analiza drzew niezdatności (FTA, ang. Fault Tree Analysis) - dedukcyjna metoda ukierunkowana na poznanie możliwych przyczyn analizowanego zdarzenia (patrz 2.3.1),
- Diagramy przyczyn/konsekwencji (CCD, ang. Cause/Consequence Diagrams) – metoda łącząca dwa rodzaje analiz w stosunku do wybranego zdarzenia; w sposób dedukcyjny odkrywa możliwe przyczyny wystąpienia zdarzenia, a w sposób dedukcyjny jego potencjalne skutki ([AR02, Nie71]),
- Analiza rodzajów, skutków i krytyczności uszkodzeń (FMECA, ang. Failure Mode, Effects, and Criticality Analysis) – metoda indukcyjna służąca przewidywaniu konsekwencji wszystkich dających się przewidzieć uszkodzeń komponentów analizowanego systemu ([MIL29, PN60812]).

2.3.1 **Analiza drzew niezdatności**

Analiza drzew niezdatności (FTA) została stworzona przez H.A. Watson z laboratoriów Bell do przeanalizowania systemu kontroli startu (ang. Launch Control System) rakiety Minuteman pod kątem nieautoryzowanego odpalenia rakiety [Vas97]. Dość szybko metoda ta znalazła uznanie firmy Boeing, potem w takich dziedzinach jak lotnictwo i energetyka atomowa. Przez pewien czas wyjątek stanowiła NASA, preferując metodę FMEA [KSW14], jednak po pożarze stanowiska startowego rakiety Apollo metoda FTA została zastosowana w tym projekcie [Eri99] po czym znalazła uznanie organizacji ([NASA02, PD-AP-1312]).

Metoda FTA jest typem dedukcyjnej analizy ukierunkowanej na zrozumienie możliwych sposobów zajścia analizowanego zdarzenia. Z reguły analizowane zdarzenie jest zdarzeniem niepożądanym. Choć możliwa jest analiza sposobów zajścia zdarzenia pozytywnego (drzewa sukcesu – ang. success trees), to doświadczenie inżynierskie wskazuje, że zazwyczaj jest znacznie więcej sposobów, na jakie system może osiągnąć sukces niż sposobów poniesienia porażki (dla przykładu: w celu analizy systemu startowego rakiety Minuteman potrzebne były tylko trzy diagramy)[NASA02].

Dochodzenie do podstawowych przyczyn zajścia wybranego zdarzenia (nazywanego zdarzeniem *szczytowym*) w metodzie FTA odbywa się poprzez określenie wszystkich zdarzeń, które mogą wystąpić w systemie i które mogą bezpośrednio skutkować zajściem zdarzenia rozważanego. Bezpośredniość rozumiana jest tu jako niewystępowanie zdarzeń pośrednich pomiędzy przyczyną, a skutkiem. Znalezione bezpośrednie przyczyny odnotowywane są na diagramie i łączone ze swoim skutkiem poprzez bramkę, która określa mechanizm powodowania zdarzenia wyjściowego. Następnie każde z tak zidentyfikowanych zdarzeń jest po kolei rozpatrywane, ustalane są jego bezpośrednie przyczyny, a zidentyfikowane zdarzenia dodawane do diagramu. Procedura ta powtarzana jest, aż do uzyskania zakładanej rozdzielczości (np. uszkodzenie pojedynczych elementów systemu jest wystarczająco szczegółowe dla zakładanych potrzeb).

Jak wspomniano wyżej, na diagramie drzew niezdatności przyczyny są łączone ze skutkami poprzez bramki. Dwa podstawowe rodzaje bramek to *AND* oraz *OR*. Bramka *AND* oznacza, że aby zaszło zdarzenie wyjściowe, muszą zajść wszystkie zdarzenia na wejściach bramki. Odpowiednio bramka *OR* oznacza, że wystarczy jedno ze zdarzeń wejściowych, aby wywołać zdarzenie na wyjściu bramki.

Tabela 2.3 Podstawowe bramki drzew niezdatności

Typ	Symbol graficzny	Znaczenia
AND		Wskazuje, że zdarzenie wyjściowe zachodzi tylko gdy wszystkie zdarzenia wejściowe zajdą.
OR		Wskazuje, że zdarzenie wyjściowe zachodzi gdy zajdzie dowolne ze zdarzeń wejściowych.
Warunkowa		Wskazuje, że zdarzenie wyjściowe zachodzi gdy zajdzie zdarzenie wejściowe (jest tylko jedno) oraz spełniony jest specjalny warunek. Warunek modelowany jest przez zdarzenie warunkujące narysowane na prawo od bramki.

W Tabela 2.3 zaprezentowano podstawowe typy bramek. Na ich wyjściu zawsze występują *zdarzenia pośrednie* (tzn. rozwijane dalej). Zdarzenia nie rozwijane dalej to *zdarzenia proste*. Jest ich kilka rodzajów. Poszczególne typy zdarzeń wraz z ich symbolami graficznymi przedstawiono w Tabela 2.4.

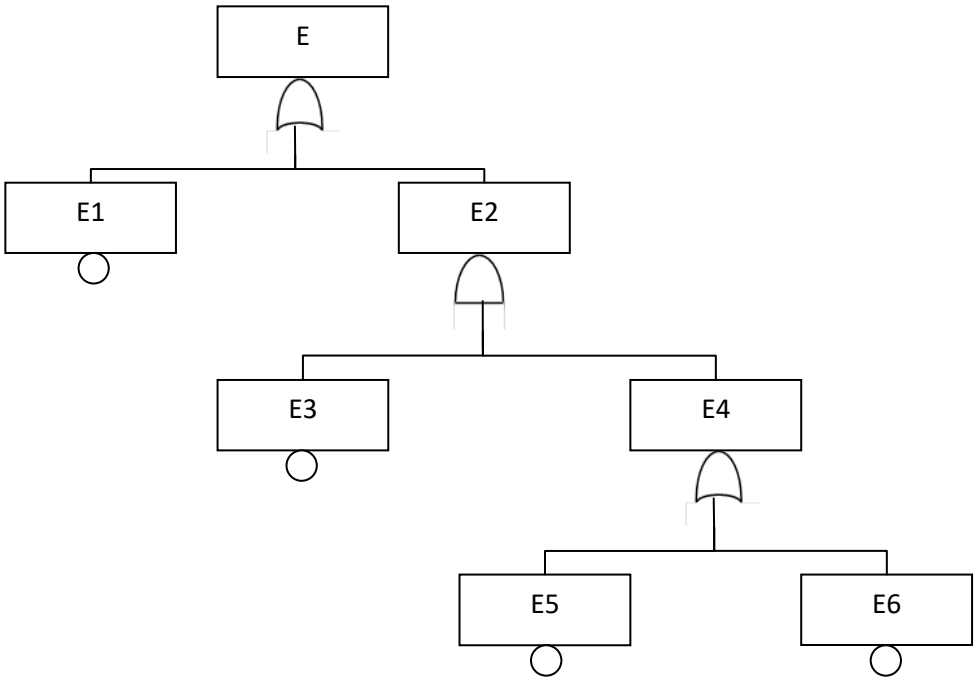
Tabela 2.4 Typy zdarzeń drzew niezdatności

Typ	Symbol graficzny	Znaczenia
Podstawowe		Zdarzenie proste, które nie wymaga dalszego rozwijania (osiągnięto zakładaną rozdzielczość).
Nierozwinięte		Zdarzenie proste, które nie zostało rozwinięte dalej z powodu braku informacji bądź ponieważ konsekwencje tego zdarzenie są nieistotne.
Zewnętrzne		Zdarzenie proste, które wystąpienie jest zakładane podczas normalnej pracy systemu, ale które może przyczynić się do wystąpienia niezdatności.
Warunkujące		Zdarzenie proste używane do określania warunków lub ograniczeń dotyczących dowolnej bramki. Używane głównie z bramką warunkową.
Rozwinięte		Zdarzenie rozwinięte dalej w osobnym drzewie niezdatności.
Pośrednie		Zdarzenie posiadające podelementy (bramkę). Zdarzenie pośrednie jest zdarzeniem wyjściowym jakiejś bramki (zdarzenie szczytowe jest specyficznym przypadkiem zdarzenia pośredniego).

Stworzone w opisany powyżej sposób drzewa niezdatności mogą być analizowane na dwa sposoby: jakościowy oraz ilościowy. Analiza jakościowa skupia się na wyznaczaniu *Minimalnych Zbiorów Przyczyn* (*MZP*).

> ***Minimalny Zbiór Przyczyn, to taki zbiór zdarzeń prostych, które doprowadzą do wystąpienia zdarzenia szczytowego drzewa niezdatności i z którego nie można usunąć żadnego zdarzenia bez naruszania poprzedniej własności.***

Dla przykładu, w drzewie na Rysunek 2.5 MZP to: $\{E1\}$, $\{E3, E5\}$, $\{E3, E6\}$.



Rysunek 2.5 Przykładowe drzewo niezdatności

Zbiory $\{E3, E5\}$, $\{E3, E6\}$ są Minimalnymi Zbiorami Przyczyn, ponieważ minimalność nie jest rozumiana jako liczba elementów (zbiór $\{E1\}$ w oczywisty sposób ma mniejszą), ale niemożliwość usunięcia ze zbioru dowolnego zdarzenia bez naruszenia właściwości umożliwiania wystąpienia zdarzenia szczytowego. A ponieważ zdarzenia *E3*, *E5* oraz *E6* samodzielnie nie spowodują wystąpienia zdarzenia *E* (bramka *AND* zdarzenia *E2*), to oba powyższe zbiory są minimalne – nie da się z nich usunąć żadnego zdarzenia tak, aby powodowały wystąpienie zdarzenia E.

Jednocześnie $\{E3, E5, E6\}$ nie jest Minimalnym Zbiorem przyczyn pomimo, że wystąpienie zdarzeń w nim zawartych spowoduje wystąpienie zdarzenia szczytowego, ponieważ po usunięciu zdarzenia *E5* bądź *E6* otrzymany zbiór będzie nadal powodował wystąpienie zdarzenia *E*.

Drugim rodzajem analizy jest analiza ilościowa. Wzbogacenie drzewa niezdatności o prawdopodobieństwa wystąpienia zdarzeń prostych umożliwia wyliczanie:

- prawdopodobieństwa wystąpienia zdarzenia szczytowego (oraz zdarzeń pośrednich),
- ważności zdarzeń – miara stopnia w którym dane zdarzenie przyczynia się do wystąpienia zdarzenia szczytowego (często się zdarza, że np. 20% zdarzeń prostych odpowiada za 90% prawdopodobieństwa wystąpienia zdarzenia szczytowego [NASA02]),
- wrażliwość – miara stopnia w jakim zmiana wartości prawdopodobieństwa wystąpienia analizowanego zdarzenia przekłada się na zmianę wartości prawdopodobieństwa wystąpienia zdarzenia szczytowego.

Powyższa lista nie jest pełną listą analiz ilościowych. Więcej informacji na ten temat można zasięgnąć w [NUREG0492].

3 Analiza drzew niezdatności z czasem

W niniejszym rozdziale przedstawiono stan badań nad drzewami niezdatności rozszerzonymi o czas, a następnie wprowadzono zarys modelu drzew niezdatności uwzględniający własności czasowe zdarzeń uczestniczących, który stanowi punkt wyjścia dla badań przedstawionych w dalszych rozdziałach pracy. Model ten został zaprezentowany w pracach [BCG91, Gor94, GW95 i War96].

3.1 Odniesienie do obecnego stanu badań

Istnieje wiele podejść do wzbogacania drzew niezdatności o informację dotyczącą dynamiki zdarzeń prowadzących do zagrożenia. Pierwszym (chronologicznie) sposobem było wprowadzenie do drzew niezdatności dynamicznych bramek. W pracy [DBB92] wprowadzono bramki CSP (cold spare), FDEP (functional dependency) oraz SEQ (sequence enforcing), a nacisk położono na analizę stochastyczną, w celu przeprowadzenia której całe drzewo niezdatności konwertowano do Łańcuchów Markowa. Potem do zestawu bramek dynamicznych dołożono AND-THEN [WM00] (zdarzenia na wejściu występują tak, że kolejne się zaczyna w momencie zakończenia poprzedniego), DDEP [MDA02] (modelowanie funkcji na żądanie), czy CCF [XMD07] (modelowanie zdarzeń o wspólnych przyczynach). Metoda ta jest rozwijana, wprowadzany jest np. dzielenie drzew niezdatności na niezależne poddrzewa, dzięki czemu można je analizować osobno (w przypadku poddrzewa statycznego można użyć algorytmu BDD do analizy) [GD97, HC07]. Wprowadzono również możliwość symulacji dynamicznych drzew niezdatności metodą Monte Carlo [MDCS98, DGS09]. Wszystkie te prace koncentrują się jednak na obliczeniach stochastycznych (możliwe jest nawet stworzenie dla wybranego drzewa niezdatności symulatora sprzętowego [AZ11]), nie umożliwiają natomiast analizowania zależności czasowych pomiędzy zdarzeniami drzew niezdatności.

Nazwa dynamiczne drzewa niezdatności użyta zostaje również w [CM02], choć technicznie jest to drzewo statyczne dostosowane do reprezentacji różnych trybów pracy analizowanego systemu (poprzez włączanie i wyłączanie różnych części drzewa). Użycie dynamicznych bramek dla opisu systemów o różnych trybach pracy opisano w [MXDO03]. W obu powyższych przypadkach informacja o dynamice systemu służy do uzyskania obliczeń statystycznych z podziałem na fazy pracy systemu.

Pokrewną do powyższej metodą analizy dynamicznego zachowania systemu w drzewach niezdatności jest użycie Sieci Petriego. Wprawdzie pierwsza praca w tym zakresie umożliwiała jedynie reprezentację statycznych drzew niezdatności [HA88], to dzięki wprowadzeniu Stochastycznych Sieci Petriego stało się możliwe również reprezentowanie dynamiki systemu [MT95]. W ramach prac nad reprezentacją drzew niezdatności przy użyciu Sieci Petriego wprowadzono do modelu elementy naprawcze (ang. repair boxes) [BR04] umożliwiające modelowanie zdarzenia naprawy systemu. W tym przypadku problemem pozostaje skupienie metody na analizie ilościowej (w celu dokonania analizy Stochastyczne Sieci Petriego konwertowane są do Łańcuchów Markowa).

Kolejnym możliwym rozszerzeniem drzew niezdatności w celu lepszej analizy dynamiki systemu jest metoda stanowo-zdarzeniowych drzew niezdatności (ang. State/Event-Fault Trees; SEFT) zaproponowana przez B. Kaisera, C. Gramlichb i M. Förster [KGF07]. Metoda ta bazuje na innym rozszerzeniu drzew niezdatności – komponentowych drzewach niezdatności (ang. Component Fault Trees; CFTs) [KLM03].

Same komponentowe drzewa niezdatności są pomyślane jako sposób powiązania drzew niezdatności ze strukturą systemu. Osiąga się to poprzez rozwinięcie idei zewnętrznych drzew niezdatności (drzew niezdatności opisujących zdarzenia zewnętrzne drzewa głównego) w koncepcję modułów. Moduł jest dowolnie wybranym wycinkiem drzewa niezdatności (w założeniu opisującym charakterystykę bezpieczeństwa/niezawodności wybranego komponent systemu), który posiada własne zdarzenia proste, bramki oraz porty wyjściowe i, co ważne, wejściowe. Użycie modułu w drzewie niezdatności wiąże się z przypisaniem jego portów wyjściowych do portów wejściowych bramek drzewa niezdatności wyższego poziomu lub innych modułów oraz jego portów wejściowych do portów wyjściowych bramek lub innych modułów drzewa niezdatności wyższego poziomu. Zaletą tego podejścia jest fakt, że moduły można ponownie używać zarówno w ramach drzew niezdatności dla różnych systemów jak również, jeśli komponent jest wykorzystany w systemie więcej niż raz, w ramach tego samego drzewa. Ponadto w razie zmiany projektu komponentu zmiany konieczne do wprowadzenia w drzewie (drzewach) niezdatności są ograniczone do odpowiadającego komponentowi modułu. Tak stworzone komponentowe drzewo niezdatności przed analizą spłaszczane jest do klasycznego drzewa niezdatności, więc możliwe do przeprowadzenia analizy są identyczne.

W metodzie stanowo-zdarzeniowych drzew niezdatności w przeciwieństwie do tradycyjnych drzew niezdatności wprowadzone zostały stany komponentów trwające przez określony czas (każdy komponent może mieć w danej chwili jeden stan) oraz natychmiastowe zdarzenia (przejście stanów jest również zdarzeniem). Metoda ta rozszerza metodę CFT poprzez dodanie możliwości modelowania komponentu przez maszynę stanów. Model ten zakłada trzy sposoby występowania zdarzeń:

- wywołanie przez inne zdarzenia,
- po deterministycznym opóźnieniu *t*,
- bądź probabilistycznym opóźnieniu (rozkład wykładniczy) od momentu wejścia w stan poprzedzający[3].

Tak przedstawiony opis dynamiki systemu zamykany jest w module, który używany jest tak jak w metodzie CFT z tą różnicą, że zarówno porty wejściowe jak i wyjściowe podzielone są na porty stanów i zdarzeń. Ponadto zależności czasowe są w tym podejściu modelowane jedynie przez bramkę opóźnienia (probabilistycznego). W celu przeprowadzenia analizy drzewo niezdatności wraz z maszynami stanów transformowane jest do postaci Sieci Petriego z Deterministycznym i Stochastycznym Czasem Aktywacji Przejść (ang. Deterministic and Stochastic Timed Transition Petri Nets; DSPN) [AMC87]. Możliwe do wyliczenia parametry analizowanego systemu zdeterminowane parametrami wyliczalnymi dla uzyskanej Sieci Petriego. W szczególności możliwe jest obliczenie prawdopodobieństwo wystąpienia wybranego stanu, jednak niekiedy może być to osiągalne jedynie drogą symulacji. Metoda ta jest rzadziej używana niż poprzednie dwie(Dynamiczne Drzewa Niezdatności oraz modelowanie drzew niezdatności przy pomocy Sieci Petriego) [MTJ14].

Wszystkie trzy powyższe metody dostosowano do prowadzenia analiz jakościowych. Analiza Sieci Petriego opisujących drzewo niezdatności umożliwia uzyskanie MCS [BFGP99], a więc informacja o

[3] Dla zdarzenia opisującego przejście stanów stan wyjściowy to stan poprzedzający a stan do którego nastąpiło przejście to stan następujący.

zachowaniu systemu jest tracona. W przypadku dynamicznych drzew niezdatności oraz SEFT możliwe jest uzyskanie Minimalnych Zbiorów Sekwencji ([TD04, XHHWZ13]). Minimalne Zbiory Sekwencji to Minimalne Zbiory przyczyn z możliwością definicji poprzedzania. A więc i w tym wypadku informacja o dynamice systemu jest skromna.

Kolejnym sposobem definiowania i wnioskowania o zależnościach czasowych w drzewach niezdatności są logiki temporalne. Choć często logiki te stosowane są do formalizacji drzew niezdatności celem weryfikacji ich kompletności i poprawności ([STR02, MNOS07]) to istnieją podejścia nastawione na analizę właściwości dynamicznych. Jedno z takich podejść przedstawiono w [Pal02], jednak jest to podejście nastawione na analizę po fakcie (wykorzystuje dziennik zdarzeń), a więc o ograniczonej przydatności w czasie projektowania systemu.

Kolejną metodą opartą o logikę temporalną jest PANDORA [WP09, WP10]. Metoda ta jest nastawiona na analizę jakościową zależności czasowych wyspecyfikowanych w drzewach niezdatności. Wynikiem tej analizy są Minimalne Zbiory Sekwencji.

W [HRS98] wykorzystywany jest Duration Calculus [CHR91]. Praca opisuje sposób w jaki pozyskać można czasowe wymagania bezpieczeństwa wobec oprogramowania. Zakłada, że wytwarzanie systemu oraz analiza bezpieczeństwa posługują się tym samym modelem systemu. Wymagania bezpieczeństwa wobec systemu formułowane są następująco: $\Box\overline{S}$ (czytaj: wszędzie zachodzi nie S), gdzie S jest wyrażeniem logicznym (w logice temporalnej) opisującym utworzone drzewo niezdatności (a więc warunkującym wystąpienie hazardu). Metoda ta dzieli zdarzenia drzewa niezdatności na takie, na które oprogramowanie ma wpływ, oraz na takie, na które nie ma. Sposób przejścia od wymagań wobec systemu do wymagań wobec oprogramowania zdefiniowany został w następujący sposób: oprogramowanie musi spełniać $\Box\overline{S}$ przy założeniu, że wszystkie zdarzenia, na które oprogramowanie nie ma wpływu są z drzewa eliminowane (zdarzenia będące na wejściu bramki *OR* nie zajdą, a będące na wejściach bramki *AND* zajdą). Dodatkowo wszystkie założenia odnoszące się do nie zajścia zdarzenia są odnotowywane.

W tym podejściu jednak specyfikowalne są jedynie długości trwania zdarzeń. Bramki drzewa nie umożliwiają określenia zależności czasowych (możliwe jest jedynie określenie kolejności wystąpienia zdarzeń za pomocą bramki *PRIORITY_AND*). Nie ma więc możliwości specyfikację zależności czasowych pomiędzy zdarzeniami i nie jest możliwe ich uzyskanie w pozyskanych wymaganiach.

Kolejną metodą włączenia informacji o dynamice systemu do analizy drzew niezdatności jest metoda Dynamicznych Grafów Przepływu (ang. Dynamic Flowgraph Methodology - DFM) [GGA95, YGA95]. W metodzie tej drzewa niezdatności budowane są w oparciu o graf przepływu zawierający przejścia czasowe. Ponieważ węzłami tego grafu mogą być wielkości fizyczne istnieje potrzeba ich dyskretyzacji przed tworzeniem drzewa niezdatności. Tworzone drzewa dzielone są na warstwy etykietowane momentem czasu; wszystkie zdarzenia w danej warstwie zachodzą we wskazanym momencie. Metoda służy do analizy systemów wbudowanych i umożliwia modelowanie zachowania zarówno oprogramowania, jak i jego środowiska. Pomimo, że metoda służy głównie do wspomagania identyfikacji przypadków testowych, możliwe jest określenie przy jej pomocy Minimalnych Zbiorów Przyczyn [YAG98]. Każdemu wydarzeniu w MZP towarzyszy informacja kiedy się wydarzyło.

Większość z powyższych podejść nie nadaje się do określania zależności czasowych w celu wywodzenia wymagań bezpieczeństwa wobec systemów komputerowych, gdyż dostarczana przez nie informacja o zachowaniu systemu prowadzącym do wystąpienia hazardu jest zbyt skromna (tylko kolejność zdarzeń). W przypadku metody Dynamicznych Grafów Przepływu problem stanowi sztywne określenie momentu wystąpienia zdarzeń w scenariuszu wystąpienia hazardu (brak możliwości specyfikacji zakresu dopuszczalnych czasów).

Jako podstawę dla proponowanej metody TREM wybrano formalizację drzew niezdatności przy użyciu notacji ECSDM [Gor94]. Opis zastosowania tej notacji w analizie drzew niezdatności można znaleźć w [GW95]. Zaproponowany sposób analizy drzew opisanych w tej notacji umożliwia uzyskanie zależności czasowych pomiędzy zdarzeniami w Minimalnych Zbiorach Przyczyn, które umożliwiają wystąpienie hazardu.

Metoda ta doczekała się rozszerzeń. Jednym z nich jest możliwość konwersji drzewa niezdatności opisanego w tej metodzie do Czasowych Sieci Petriego [GMA95]. Konwersja taka daje możliwość przeprowadzenia analizy osiągalności hazardu[GW97], jednak wymusza utworzenie słownika dostępnych bramek i ich sieciowych odpowiedników (w notacji ECSDM istnieje duża swoboda określania zależności czasowych).

Konwersja ta leży również u podstaw powstania metody FTTD. W [MS00, MS02] pokazano jak można efektywnie obliczyć zależności czasowe w drzewach niezdatności, jeśli bramki drzewa należą do przywołanego wyżej słownika. Aby uzyskać taki rezultat trzeba było jednak iść na kompromis. Zależności czasowe w Minimalnych Zbiorach Przyczyn wyliczonych w ten sposób odnoszą się zawsze do wyróżnionego punktu w czasie (początek zdarzenia szczytowego). Tracone są relacje czasowe pomiędzy zdarzeniami prostymi w drzewie. Tak obliczone dane dobre są więc do określania czasu reakcji systemów zabezpieczających, natomiast nie nadają się do określania zależności czasowych pomiędzy zdarzeniami inicjującymi.

Opisywana metoda została użyta do analizy zabezpieczeń linii energetycznych [LMS11], zależności czasowych w systemie logistycznym [MNSW08] czy systemu Ruchomego Odstępu Blokowego [MLST12]. Wprowadzono też do modelu aspekty probabilistyczne [BLM09, BLM10]. Nie wprowadzono jednak dotychczas automatyzacji przetwarzania MZP z zależnościami czasowymi celem wyodrębnienia wymagań czasowych wobec systemu.

Zaproponowana w tej pracy metoda TREM skupia się na analizie zależności czasowych pomiędzy zdarzeniami z MZP. Wyróżnia ją to na tle większości metod wprowadzających opis dynamiki systemu do drzew niezdatności, gdyż te wykorzystują elementy dynamiczne do rozbudowywania możliwości analizy stochastycznej. W większości z tych metod analiza zależności czasowych w MZP ogranicza się do określenia kolejności występowania zdarzeń.

Wyjątkiem jest metoda formalizacji drzew niezdatności przy użyciu notacji ECSDM. Pozwala ona na wyznaczanie skomplikowanych zależności czasowych w MZP przy użyciu algorytmu TGRAF-MZP. Metoda ta nie wspomaga jednak wyznaczania wymagań bezpieczeństwa, a także pozostawia pole do ulepszeń w zakresie optymalizacji wykorzystywanych algorytmów. Rozwijająca te podejście metoda FTTD oferuje znaczący wzrost wydajności, jednak kosztem utraty informacji umożliwiających

określenie czasowych wymagań bezpieczeństwa wobec analizowanego systemu (metoda skupia się na określaniu wymagań czasowych wobec systemów zabezpieczających).

Jedynie Hansen et al. proponuje metodę wywodzenia wymagań czasowych na podstawie analizy drzew niezdatności. Ponieważ w tej metodzie używany jest Duration Calculus, wywodzone wymagania mogą odnosić się jedynie do czasu trwania pojedynczych zdarzeń oraz kolejności ich występowania po sobie. Metoda TREM nie posiada takich ograniczeń i umożliwia określanie chociażby czasu łącznego występowania dwóch zdarzeń, czasu trwania sumy dwóch zdarzeń, czy odstępu w czasie pomiędzy zdarzeniami.

Podsumowując, metoda TREM wyróżnia się na tle innych metod wzbogacania analizy drzew niezdatności informacjami o dynamice systemu poprzez zaproponowanie wywodzenia szerokiego spektrum wymagań czasowych wobec analizowanego systemu.

3.2 Formalna definicja zachowania systemu

Punktem wyjścia do zaproponowanej w tej pracy metody są modele i algorytm przedstawione w pracach [BCG91, Gor94, GW95 i War96]. Przedstawione w nich modele zostają (z rozszerzeniami) przeniesione do zaproponowanej metody. Zostaną one opisane w dalszej części rozdziału.

Do formalnego modelowania zdarzeń w analizowanym systemie stosowana jest notacja ECSDM [Gor94]. Jej zastosowanie do analizy drzew niezdatności zostało zaprezentowane w [GW95].

Notacja ta składa się z dwóch komponentów:

- modelu dynamicznego, służącego do opisu zachowania systemu,
- modelu statycznego, umożliwiającego wyspecyfikowanie elementów systemu i ich możliwych stanów.

Model dynamiczny składa się z następujących elementów:

E - zbiór *zdarzeń*, którego elementy oznaczamy jako X, Y, Z.

L - zbiór *etykiet* używanych do identyfikacji wystąpień zdarzeń; etykiety są oznaczane przez *l*, *m*, *n*.

A - zbiór *akcji* – zdarzeń opatrzonych etykietami (różne etykiety umożliwiają rozróżnienie różnych wystąpień tego samego zdarzenia) oraz wyróżnionej akcji "cichej" (oznaczonej $\perp$) trwającej przez cały okres życia systemu,

(i) $A = \{L \times E\} \cup \{\perp\}$.

Akcje są oznaczane przez *x*, *y*, *z*.

ϕ - funkcja $E \to \mathbb{P}(A)$, która dla zadanego zdarzenia X zwraca wszystkie akcje tego zdarzenia:

(ii) $\phi(X) \stackrel{\text{def}}{=} \big((l, X)\big| l \in L\big)$.

Funkcja ϕ może być etykietowana wybraną funkcją *Time* (patrz niżej) i wtedy zwraca akcje danego zdarzenia zachodzące w zachowaniu systemu opisanym zadaną funkcją *Time*.

T - zbiór *tranzycji* będących początkami bądź końcami wybranej akcji. Tranzycje oznaczane są przez symbol akcji wraz z odpowiednim indeksem: $_s$ (początek akcji) lub $_e$ (koniec akcji). Zbiór *T* składa się z:

(iii) $T = \{x_s | x \in A\} \cup \{x_e | x \in A\}$

$\prec_c$ - relacja przyczynowości na zbiorze $(T \times T)$. Jest to asymetryczna i przechodnia relacja porządku częściowego.

$=_c$ - relacja równości przyczynowej na zbiorze $(T \times T)$. Zachodzi pomiędzy tranzycjami o takich samych przyczynach.

Dla każdej tranzycji w prawdziwe są wyrażenia:

$\perp_s \prec_c w \ \vee \perp_s =_c w$ oraz $w \prec_c \perp_e \vee w =_c \perp_e$.

$\mathbb{R}$ - zbiór liczb rzeczywistych.

Time - funkcja częściowa $T \to \mathbb{R}$ przypisująca tranzycjom czas ich wystąpienia. Time może być interpretowana jako zbiór par (w, r), gdzie $r \in \mathbb{R}$, $w \in T$ i $Time(w) = r$. Różne funkcje *Time* reprezentują różne zachowania opisywanego systemu. Każda funkcja *Time* musi spełniać następujące warunki:

- jeśli *w* i $w' \in \boldsymbol{dom}\, Time$ to:

  (iv) $w \prec_c w' \Longrightarrow Time(w) < Time(w')$, oraz
  (v) $w =_c w' \Longrightarrow Time(w) = Time(w')$.

- $\perp_s \in \boldsymbol{dom}\, Time$ oraz $\perp_e \in \boldsymbol{dom}\, Time$.

start - funkcja częściowa $A \to \mathbb{R}$, która dla danej akcji *x* i zadanego *Time* zwraca r takie, że $(x_s, r) \in Time$,

end - funkcja częściowa $A \to \mathbb{R}$, która dla zadanego *x* i *Time* zwraca r takie, że $(x_e, r) \in Time$.

mk-action - funkcja $(Time \times Time) \to A$ umożliwiająca definiowanie akcji na podstawie dwóch tranzycji i czasów ich wystąpień. Dla wszystkich tranzycji i akcji występuje następująca zależność:

(vi) $\forall x \in A \cdot \exists x_s, x_e \in T, r, r' \in \mathbb{R} \cdot x = mk - action\big((x_s, r), (x_e, r')\big) \Leftrightarrow (x_s, r), (x_e, r') \in Time$

$\prec_t$ - relacja uporządkowania czasowego na zbiorze $(Time \times Time)$. Jest relacją antysymetryczną i przechodnią zdefiniowaną jako:

(vii) $\forall t, t' \in Time \cdot t = (w, r) \wedge t' = (w', r') \Rightarrow (t \prec_t t' \Leftrightarrow w \prec_c w' \wedge r < r')$

$=_t$ - relacja równości czasowej na zbiorze $(Time \times Time)$ zdefiniowana jako:

(viii) $\forall t, t' \in Time \cdot t = (w, r) \wedge t' = (w', r') \Rightarrow (t =_t t' \Leftrightarrow w =_c w' \wedge r = r')$

$\prec_h$ - relacja przyczynowości "head". Oznacza ona, że akcja będąca drugim argumentem relacji jest spowodowana przez początek akcji będącej pierwszym argumentem. Można to zdefiniować jako:

(ix) $$\forall x, y \in A \cdot x = mk - action(t_s, t_e) \wedge y = mk - action(u_s, u_e) \Rightarrow (x \prec_h y \Leftrightarrow t_s \prec_t u_s)$$

$\prec_i$ - relacja przyczynowości "interior". Oznacza ona, że akcja będąca drugim argumentem relacji jest spowodowana przez koniec akcji będącej pierwszym argumentem. Można to zdefiniować jako:

(x) $$\forall x, y \in A \cdot x = mk - action(t_s, t_e) \wedge y = mk - action(u_s, u_e) \Rightarrow (x \prec_i y \Leftrightarrow t_e \prec_t u_s)$$

Należy zauważyć, że *start*, *end*, *mk-action*, $\prec_t$, $=_t$, $\prec_h$and $\prec_i$ zależą od funkcji $\boldsymbol{Time}$. Ponadto, jako że $\boldsymbol{Time}$ określa konkretne zachowanie systemu, jeśli jakaś tranzycja nie należy do dziedziny $\boldsymbol{Time}$, to oznacza, że nie występuje w tym zachowaniu systemu.

3.3 Formalny model budowy systemu

Notacja ECSDM w celu statycznego opisu systemu posługuje się notacją wzorowaną na notacji VDM [Jon90], składającą się z następujących elementów:

types

definicja typów

values

wartości stałe

state SYSTEM of

element1: zbiór stanów elementu element1

element2: zbiór stanów elementu element2

element3: zbiór stanów elementu element3

…

Na podstawie stanów elementów systemu definiowane są zdarzenia. Do ich definicji służy notacja:

(xi) $$E(Time, t) \equiv PE/(Time, t),$$

która oznacza, że w zachowaniu systemu określonym przez *Time* zdarzenie *E* występuje w momencie *t* wtedy i tylko wtedy, gdy predykat *PE* jest spełniony w momencie *t*.

$\boldsymbol{PE}$ jest nazwany *predykatem charakterystycznym*.

Jeżeli predykat $\boldsymbol{PE}$ jest zdefiniowany przez stan pojedynczego elementu systemu[4], wyspecyfikowane zdarzenie nazywamy *prostym*. Zdarzenie *SE* jest takim zdarzeniem:

(xii) $SE(Time, t) \equiv (element1 = stan3)/(Time, t).$

Zdarzenia *złożone* odnoszą się do stanu więcej niż jednego elementu systemu. Definicja takiego zdarzenia może mieć formę:

(xiii) $E(Time, t) \equiv (PE1 \wedge PE2)/(Time, t),$

gdzie predykaty *PE1* i *PE2* są predykatami zdarzeń prostych. Oczywiście zdarzenia złożone mogą być wyspecyfikowane przez stan więcej niż dwóch elementów systemu.

Ostatnim typem zdarzenia jest zdarzenie *czasowe*, którego definicja odnosi się bezpośrednio do czasu. Zdarzenia takie specyfikuje są jako:

(xiv) $E(Time, t) \equiv \exists e \in \phi_{Time}(E') \cdot PT(e, t),$

gdzie *E'* jest zdarzeniem nieodnoszącym się do czasu (prostym lub złożonym), a predykat *PT* definiuje relację czasową pomiędzy zdarzeniami *E* i *E'*.

3.4 Specyfikacja zależności czasowych w drzewach niezdatności

Definicja drzewa niezdatności rozszerzonego o czas polega na wzbogaceniu standardowego modelu o definicje zależności czasowych dla poszczególnych bramek. Każda bramka jest zdefiniowana za pomocą funkcji semantycznej M, która ma następującą postać:

(xv) $\mathrm{M}\left(\mathrm{G_j}\left(\mathrm{OUTPUT_j}\in\mathrm{E},\mathrm{INPUT_j}\subseteq\mathrm{E}\right)\right) \equiv \forall_{o\in\phi(OUTPUT_j)} occur(o) \Longrightarrow \bigvee_{k=1}^{m}\left(\bigwedge_{l=1}^{n}\left(\exists i_{kl} \in \phi(I_{kl}) \cdot I_{kl} \in INPUT_j \wedge occur(i_{kl})\right) \wedge enabling\_condition_k(o, i_{k1}, \cdots, i_{kn})\right)$[5],

gdzie $enabling\_condition$ to wyrażenie logiczne w notacji ECSDM opisujące zależności czasowe pomiędzy zadanymi akcjami.

Funkcja ta definiuje bramkę $\mathrm{G_j}$ o zdarzeniu wyjściowym $\mathrm{OUTPUT_j}$ i zbiorze zdarzeń wejściowych $\mathrm{INPUT_j}$. Ponadto stanowi, że aby wystąpiła akcja zdarzenia $\mathrm{OUTPUT_j}$ musi wystąpić zestaw akcji zdarzeń wejściowych wyspecyfikowanych w jednym z m wyrażeń (indeksowanych zmienną k) oraz, że pomiędzy tymi akcjami musi zajść zależność czasowa określona odpowiednią funkcją $enabling\_condition_k$. Funkcja $enabling\_condition_k$ opisuje również zależność pomiędzy akcjami zdarzeń wejściowych a akcją zdarzenia wyjściowego.

Przykład 1.

Dla przykładu definicja bramki $\mathrm{G_{AND}}$ posiadającej zdarzenie wyjściowe E i zdarzenia wejściowe E_1 oraz E_2 mogłaby wyglądać następująco:

[4] Zdarzenie proste notacji ECSDM nie jest tożsame ze zdarzeniem prostym drzewa niezdatności. Jeśli nie podano inaczej, odwołania do zdarzeń prostych w tej pracy oznaczają zdarzenia proste drzewa niezdatności.

[5] Dla wygody notacji w stosunku do oryginału zmieniono listę zdarzeń wejściowych na zbiór.

(xvi) $\mathrm{M}\big(\mathrm{G}_{\mathrm{AND}}(\mathrm{E},\{E_1,E_2\})\big) \equiv \forall_{e\in\phi(E)} occur(e) \Longrightarrow \exists e_1 \in \phi(E_1), e_2 \in \phi(E_2) \cdot occur(e_1) \wedge occur(e_2) \wedge overlap(e_1,e_2) \wedge start(e) = max\big(start(e_1), start(e_2)\big) \wedge end(e) = min\big(end(e_1), end(e_2)\big)$

Funkcja (xvi) definiuje zdarzenie wyjściowe bramki jako wspólne występowanie zdarzeń wejściowych: a więc wystąpienia zdarzeń wejściowych muszą się pokryć, a zdarzenie wyjściowe rozpoczyna się wraz z początkiem późniejszego zdarzenia wejściowego i kończy wraz z końcem zdarzenia wejściowego, które zakończy się najpierw. Ponieważ definiowana jest bramka *AND* w definicji występuje jedno *enabling_condition* o postaci:

(xvii) $enabling\_condition(e,e_1,e_2) = \big(overlap(e_1,e_2) \wedge start(e) = max\big(start(e_1),start(e_2)\big) \wedge end(e) = min\big(end(e_1),end(e_2)\big)\big)$.

W przypadku definiowania bramki *OR* występowałoby tyle *enabling_condition* ile byłoby alternatyw.

3.5 Analiza drzewa niezdatności z czasem

Praca [War96] wprowadza algorytm (nazwany TGRAF-MZP) umożliwiający analizę zależności czasowych w drzewach niezdatności opisanych przy pomocy przedstawionej powyżej notacji oraz wyznaczanie zależności czasowych dla poszczególnych Minimalnych Zbiorów Przyczyn. Algorytm ten nie jest częścią proponowanej w tej pracy metody, ale ponieważ zaproponowane algorytmy zostały o niego oparte, zostanie on przedstawiony poniżej.

Algorytm TGRAF-MZP reprezentuje zależności czasowe występujące w drzewach niezdatności w postaci tzw. T-grafu. T-graf jest to graf skierowany z krawędziami, którym przypisano wagi składające się z dwóch wartości:

- długości będącej liczbą rzeczywistą,
- relacji (przyjmującą wartość $>$ bądź $\geq$).

Krawędź jest *cięższa*, jeśli jej długość jest większa, bądź, przy tej samej długości, związana z nią relacja to $\geq$ podczas gdy relacja przypisana krawędzi z nią porównywanej to $>$ (odpowiednio definiowane jest pojęcie wagi *lżejszej* oraz wag *równych).*

Zdefiniowano również operację dodawania wag przypisanych krawędziom T-grafu. Waga wynikowa ma długość będącą sumą długości wag składowych. Natomiast przypisana jej relacja to $\geq$ jeśli relacje obu wag składowych to $\geq$, przeciwnym razie przypisana jej relacja to $>$.

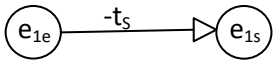
Określone zostały również dwie właściwości T-grafów:

- T-graf pełny – jest to graf, w którym dla każdej pary wierzchołków połączonych co najmniej jedną ścieżką istnieje bezpośrednia krawędź o wadze równej wadze najlżejszej z tych ścieżek,
- T-grafy zgodne – są to grafy o takim samym zbiorze wierzchołków, które mają takie same wagi najlżejszych ścieżek dla wszystkich par wierzchołków.

Algorytm TGRAF-MZP wykorzystuje algorytm pomocniczy PTGRAF przekształcający zadany T-graf w zgodny z nim T-graf pełny.

Aby go zastosować należy przekształcić definicje bramek w odpowiadający im T-graf. Przebiega to następująco:

1. Wyrażenia czasowe w definicji bramki przekształcane są do zbioru porównań czasów wystąpień tranzycji zdarzeń (tzw. postać normalna ECSDM) zawartych w definicji (z możliwym przesunięciem czasowym), np. $duration(e_1) \geq t_s$ przekształcane jest do postaci $start(e_1) + t_s \leq end(e_1) \Leftrightarrow end(e_1) - t_s \geq start(e_1)$. Praca [War96] zawiera słownik wyrażeń normalnych ECSDM odpowiadających poszczególnym wyrażeniom ECSDM.
2. Czasy wystąpienia tranzycji zdarzeń wejściowych i wyjściowych (nieznane) reprezentowane są jako wierzchołki T-grafu.
3. Poszczególne wyrażenia w postaci normalnej reprezentowane są jako krawędzie między wierzchołkami, np.: $end(e_1) - t_s \geq start(e_1)$ jest reprezentowane jako:



gdzie pusty grot reprezentuje nierówność nieostrą.

Przykład 2.

Dla przykładu wyrażenie

$$occur(x_1) \wedge occur(x_2) \wedge occur(x_3) \wedge duration(x_1, x_2) > 4 \wedge end(x_2) + 3 < start(x_3)$$

jest reprezentowane następująco:

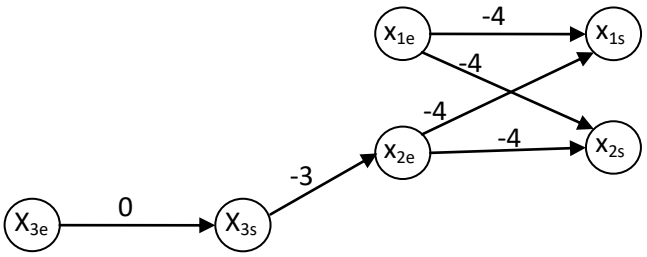
$$start(x_1) < end(x_1) \wedge start(x_2) < end(x_2) \wedge start(x_3) < end(x_3) \wedge start(x_1) + 4 < end(x_1) \wedge start(x_1) + 4 < end(x_2) \wedge start(x_2) + 4 < end(x_2) \wedge start(x_2) + 4 < end(x_1) \wedge end(x_2) + 3 < start(x_3)$$

gdzie pierwsze trzy porównania wynikają z predykatu $occur(\cdot)$, następne cztery zaś z predykatu $duration(\cdot,\cdot)$.

Wyrażenie to może zostać zredukowane do postaci

$$start(x_3) < end(x_3) \wedge start(t_1) + 4 < end(t_1) \wedge start(t_1) + 4 < end(t_2) \wedge start(t_2) + 4 < end(t_2) \wedge start(t_2) + 4 < end(t_1) \wedge end(x_2) + 3 < start(x_3),$$

jako że porównania 1-wsze i 2-gie są ogólniejsze od odpowiednio 4-tego i 6-tego (i mogą zostać pominięte bez utraty informacji). Ostatecznie wyrażenie to w postaci T-grafu będzie miało postać:



gdzie zapełniony grot reprezentuje nierówność ostrą.

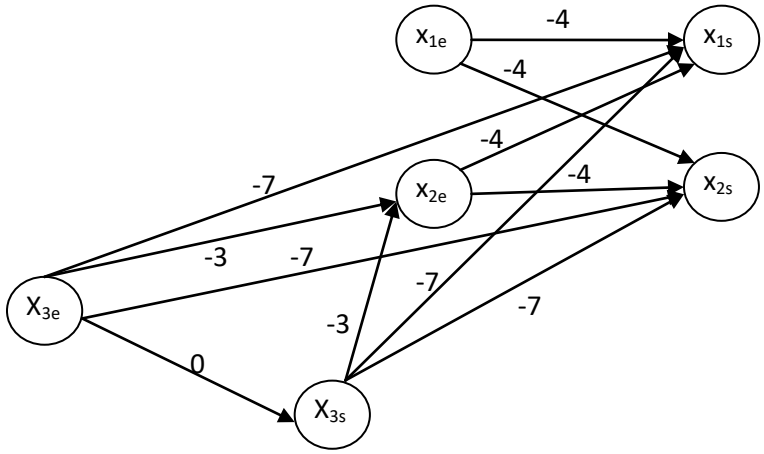
Na tak zbudowanym grafie operuje algorytm PTGRAF. W skrócie można powiedzieć, że algorytm ten wyszukuje ścieżki dwuelementowe i tworzy krawędź z początku do końca ścieżki. Jako wagę tworzonej krawędzi przyjmuje sumę wag (dwóch) krawędzi ścieżki i dodaje taką krawędź do grafu jeśli:

a) pomiędzy wierzchołkami nie ma jeszcze krawędzi,
b) istniejąca krawędź ma wyższą wagę - jest zastępowana (zasadność tej operacji w kontekście zależności czasowych rozważono m.in. w [Gol06]).

Operacja ta powtarzana jest dopóki nie da się już dodać do grafu żadnej nowej krawędzi (w wyszukiwaniu uwzględniane są też krawędzie nowo dodane).

Przykład 3.

W efekcie zastosowania algorytmu PTGRAF na grafie z poprzedniego przykładu powstaje graf pełny, który zobrazowano poniżej.



Jak widać, w efekcie zastosowania algorytmu dodanych zostało 5 krawędzi.

Natomiast algorytm TGRAF-MZP jest stosowany w stosunku do wybranego drzewa niezdatności i przebiega następująco:

1. Obliczane są MZP metodą tradycyjną.
2. Dla każdego MZP:

a. wyznaczane jest minimalne (w sensie liczby bramek) poddrzewo drzewa niezdatności zawierające zdarzenie szczytowe i zdarzenia elementarne składające się na konkretny MZP,
b. poddrzewo te przekształca się do postaci T-grafu (T-grafy poszczególnych bramek łączy się poprzez wierzchołki o tych samych etykietach),
c. na tak powstałym T-grafie wykonuje się algorytm PTGRAF, który przekształci zadany T-graf w zgodny T-graf pełny, ujawniając niejako informacje o zależnościach czasowych wyrażone implicite[6],
d. z powstałego grafu odrzucane są wierzchołki nie reprezentujące tranzycji zdarzeń z MZP oraz wszystkie krawędzie z nimi związane,
e. pozostałe wyrażenia przekształca się do postaci ECSDM i stanowią one $enabling\_condition$ dla analizowanego MZP.

[6] Dla wyrażenia $a + 3 < b \wedge b + 2 \leq c$ informacje wyrażone explicite to $a + 3 < b$ oraz $b + 2 \leq c$. Natomiast informacja wyrażona implicite to $a + 5 < c$.

4 Metoda TREM wyznaczania wymagań bezpieczeństwa

W tym rozdziale przedstawiono nową metodę TREM (*Timing Requirements sElection Method*) wyznaczania czasowych wymagań bezpieczeństwa poprzez analizę drzew niezdatności.

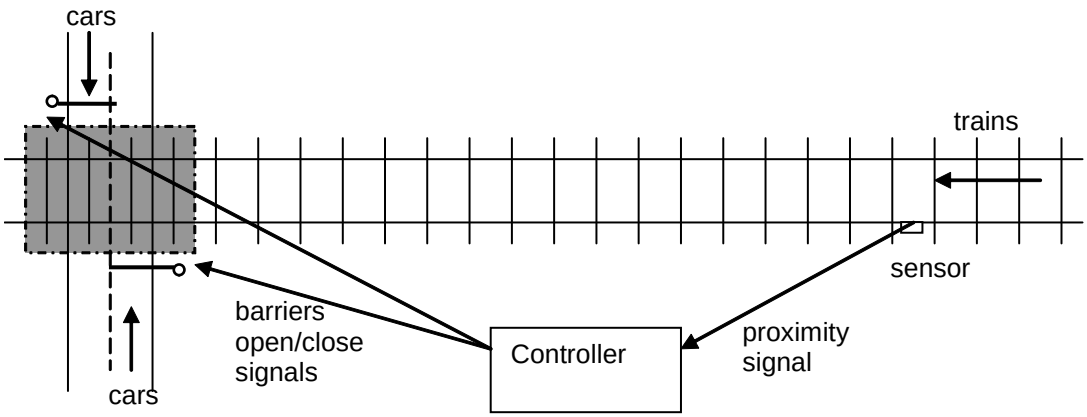
4.1 Ogólna idea metody

Ideą metody jest zaproponowanie, na podstawie analizy zależności czasowych określonych w drzewie niezdatności objętego zainteresowaniem systemu, dodatkowych wymagań czasowych wobec tego systemu. Spełnienie tych wymagań uniemożliwia materializację zagrożenia analizowanego w drzewie niezdatności nawet wtedy, gdy zmaterializują się wszystkie zdarzenia z jednego lub większej liczby Minimalnych Zbiorów Przyczyn (MZP). W ten sposób metoda TREM prowadzi do zablokowania kluczowych zagrożeń (hazardów) w sytuacjach, które tradycyjna analiza drzew niezdatności traktuje jako odblokowujące wystąpienie hazardu.

Punktem wyjścia metody jest klasyfikacja zdarzeń prostych występujących w drzewie niezdatności. Zdarzenia te są klasyfikowane jako zdarzenia kontrolowane, przewidywalne, obserwowalne i nieobserwowalne. Należy podkreślić, że jest to klasyfikacja subiektywna, dokonywana z perspektywy systemu sterującego (a więc dla dwóch różnych systemów sterujących ta klasyfikacja może wyglądać różnie). Wyjaśnienie tej klasyfikacji podano w podrozdziale 4.3.1.

Przykład 4.

Jako przykład rozważmy strzeżony przejazd dwukierunkowej drogi kołowej przez jeden tor kolejowy przedstawiony na Rysunek 4.1(opis systemu można znaleźć w podrozdziale 9.1.2.2).



Rysunek 4.1 Diagram przejazdu kolejowego

W systemie tym zdarzeniami kontrolowalnymi są wystąpienia sygnałów otwarcia i zamknięcia zapór drogowych (inicjuje je wbudowany system sterujący, który decyduje o ich parametrach czasowych). Zdarzeniem przewidywalnym jest wjazd pociągu na przejazd – system sterujący wie o nim z wyprzedzeniem dzięki sygnałowi z czujnika umieszonego na torach. Sam sygnał z czujnika (sygnał o nadjeżdżającym pociągu) jest zdarzeniem obserwowalnym – system sterujący go

zaobserwuje, ale nie może przewidzieć. Zdarzeniami nieobserwowalnymi są wjazdy pojazdów drogowych na teren skrzyżowania (rozpatrywany system nie jest wyposażony w odpowiednie czujniki wykrywające pojazdy drogowe). Każde zdarzenie należy do dokładnie jednej kategorii, a zdarzenia klasyfikujemy w odniesieniu do momentu, w którym dochodzi do wystąpienia (materializacji) zdarzenia. Jeśli system sterujący wie o danym zdarzeniu z wyprzedzeniem (w stosunku do momentu jego wystąpienia), to zdarzenie jest przewidywalne, jeśli dowiaduje się o nim dopiero w momencie wystąpienia, to obserwowalne. Tak więc dodanie drugiego czujnika obecności pociągu, tym razem na skrzyżowaniu, nie spowoduje, że wjazd pociągu na przejazd stanie się zdarzeniem obserwowalnym (ponieważ ten wjazd zostanie przewidziany na podstawie sygnału z czujnika umieszczonego na torach). Nie oznacza to oczywiście, że dodanie czujnika wjazdu pociągu na przejazd jest bezcelowe. W powyższym przypadku poprawi to precyzję określenia momentu wystąpienia zdarzenia, gdyż dokładny czas wjazdu pociągu na przejazd zależy od jego prędkości. Niemniej taka modyfikacja rozważanego systemu nie zmienia klasyfikacji zdarzeń prostych (zgodnie z powyższą definicją zdarzenie, którego wystąpienie da się zaobserwować jest obserwowalne jedynie wtedy, gdy nie jest przewidywalne). Powyższe rozważania dotyczą istoty poszczególnych typów zdarzeń, natomiast ich klasyfikacja za każdym razem dokonywana jest przez analityka.

Metoda TREM jest oparta na obserwacji, że możliwe jest wywarcie wpływu na zależności czasowe pomiędzy zdarzeniami kontrolowalnymi a zdarzeniami innymi niż nieobserwowalne. dzięki czemu można uniknąć niektórych scenariuszy wystąpienia zagrożenia.

Przykład 5.

Dla systemu przedstawionego na Rysunek 4.1 jako zagrożenie przyjmijmy jednoczesne przebywanie w rejonie skrzyżowania pociągu i pojazdu drogowego.

Rozważmy następujące scenariusze:

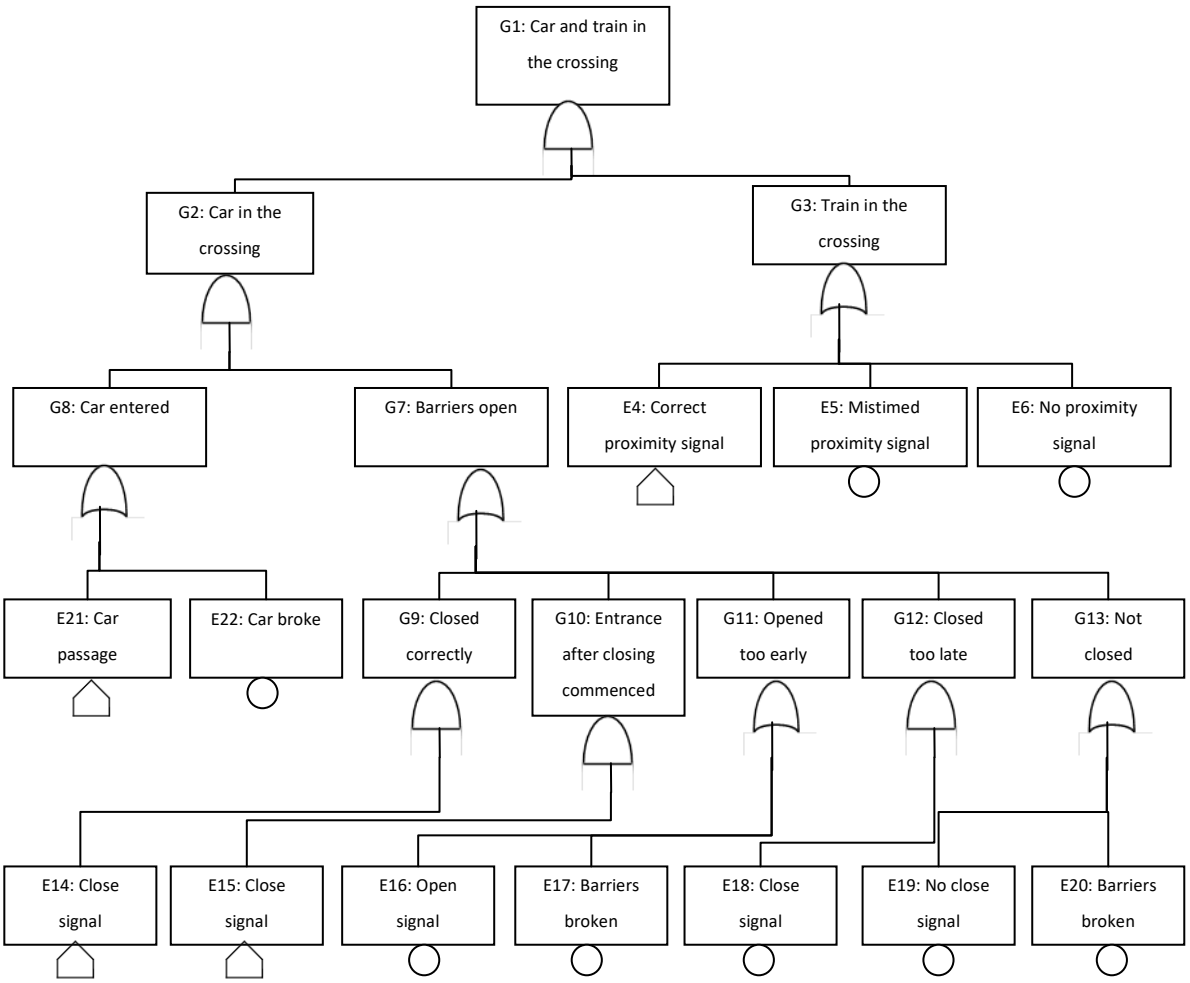
S1: wysłanie sygnału zamknięcia zapór drogowych przed wjazdem pociągu na przejazd,

S2: wysłanie sygnału zamknięcia zapór drogowych po sygnale o nadjeżdżającym pociągu.

Powyższe scenariusze dotyczą w istocie tej samej akcji (zamknięcie zapór drogowych) opisanej w odniesieniu do dwu różnych zdarzeń: *wjazdu pociągu na przejazd* oraz *sygnału o nadjeżdżającym pociągu*. Zauważmy, że skuteczność tej akcji zależy od parametrów czasowych zdarzeń, do których odwołujemy się w powyższych scenariuszach.

W scenariuszu S1 wyprzedzenie, z jakim należy wysłać sygnał zamknięcia zapór zależy od czasu reakcji samych zapór na sygnał sterujący (opóźnienie od wysłania sygnału do rozpoczęcia opuszczania zapór) oraz czasu jaki potrzebuje samochód na opuszczenie przejazdu (jeśli założyć, że samochody będą wjeżdżały na przejazd

nawet po rozpoczęciu opuszczania zapór drogowych, to do wyprzedzenia z jakim wysłany powinien być sygnał zamknięcia zapór należy doliczyć również czas opuszczania zapór). Natomiast w scenariuszu S2 maksymalne opóźnienie w wysłaniu sygnału zamknięcia zapór (w stosunku do wystąpienia sygnału o nadjeżdżającym pociągu) można określić jako różnicę pomiędzy minimalnym czasem pomiędzy sygnałem o nadjeżdżającym pociągu a wjazdem pociągu na przejazd oraz minimalnym wyprzedzeniem z jakim należy wysłać sygnał zamknięcia zapór przed wjazdem pociągu na przejazd (aby umożliwić samochodom opuszczenie przejazdu).[7]



Rysunek 4.2 Drzewo niezdatności dla systemu przejazdu kolejowego

W drzewie niezdatności stworzonym dla rozpatrywanego systemu (patrz Rysunek 4.2) odwoływano się do sygnałów otwarcia i zamknięcia zapór oraz sygnału o

[7] Przykład ten wyjaśnia również motywację do wprowadzenia podziału zdarzeń na obserwowalne i przewidywalne. Określając wymagania wobec systemu można żądać reakcji na wystąpienie zdarzenia obserwowalnego dopiero po jego zaistnieniu (scenariusz S2), natomiast przy zdarzeniu przewidywalnym można żądać zastosowania działań wyprzedzających – mających nastąpić przed wystąpieniem rozważanego zdarzenia (scenariusz S1).

zbliżającym się pociągu. Nie odnoszono się natomiast do zdarzenia wjazdu pociągu na skrzyżowanie (nie jest to zdarzenie proste).

Metoda TERM identyfikuje takie zależności czasowe między sygnałami otwarcia i zamknięcia zapór a sygnałem o zbliżającym się pociągu, które umożliwiają uniknięcie niektórych scenariuszy wystąpienia zagrożenia prezentowanych w powyższym drzewie niezdatności.

Przykład 6.

Przykładem takiego dodatkowego wymagania, które zostało zidentyfikowane z wykorzystaniem metody TERM dla rozpatrywanego przykładu jest:

$$\forall e_4 \in \phi(E_4) \cdot \exists e_{15} \in \phi(E_{15}) \cdot start(e_4) - T_d - T_{bmax} - T_{emax} + T_{cmin} \geq start(e_{15}) \Leftrightarrow \forall e_{15} \in \phi(E_{15}) \cdot \exists e_4 \in \phi(E_4) \cdot start(e_4) + 33 \geq start(e_{15})$$

Oznacza ono, że aby uniknąć zagrożenia nawet w przypadku samochodów wjeżdżających na przejazd po rozpoczęciu zamykania zapór, sygnał zamknięcia zapór musi być wysłany nie później niż 33 jednostki czasu po odebraniu sygnału o nadjeżdżającym pociągu. Czas ten jest warunkowany następującymi stałymi czasowymi: T_d - czas od wysłania sygnału sterującego do rozpoczęcia zamykania zapór, T_{bmax} – maksymalny czas zamykania zapór drogowych, T_{emax} – założony maksymalny czas przejazdu samochodu przez przejazd kolejowy, T_{cmin} - minimalny czas upływający pomiędzy wykryciem pociągu, a jego wjazdem na skrzyżowanie.

4.2 Definicja metody

Metoda TREM na wejściu przyjmuje tradycyjne drzewo niezdatności stworzone dla systemu i składa się z następujących kroków:

Krok 1. Klasyfikacja zdarzeń prostych występujących w drzewie niezdatności jako zdarzeń kontrolowalnych, obserwowalnych, nieobserwowalnych i przewidywalnych (patrz podrozdział 4.3.1).

Krok 2. Określenie zależności czasowych dla poszczególnych bramek drzewa (patrz podrozdział 3.4).

Krok 3. Opcjonalne określenie dodatkowych ograniczeń dziedzinowych dotyczących analizowanego systemu (patrz rozdział 7).

Krok 4. Identyfikacja Minimalnych Zbiorów Przyczyn oraz analiza zależności czasowych dla każdego z nich (rozdział 6).

Krok 5. Określenie kandydujących wymagań czasowych wobec systemu sterującego (podrozdział 7.1).

Krok 6. Wybór dodatkowych wymagań czasowych wobec systemu związanych z zapewnianiem bezpieczeństwa.

Kroki 4. oraz 5. są przeprowadzane automatycznie. Natomiast kroki 1, 2 , 3 i 6 wymagają udziału analityka. Wynikiem metody jest zbiór wyrażeń, które specyfikują dodatkowe wymagania odnoszące się do zdarzeń i ich charakterystyk czasowych. Wyrażenia te odwołują się do zdarzeń kontrolowalnych, a więc stanowią dodatkowe wymagania, które mogą być zaimplementowane w systemie sterującym. Implementacja tych wymagań uniemożliwia materializację niektórych

scenariuszy zagrożeń[8] opisanych w drzewie niezdatności. W ten sposób zastosowanie metody przyczynia się do podniesienia bezpieczeństwa systemu.

Metoda TERM nie obejmuje sposobu implementacji i weryfikacji spełnienia określonych przy jej pomocy wymagań czasowych. Implementacja tych dodatkowych wymagań może być dokonana na przykład poprzez odpowiednią modyfikację algorytmu sterownika, 'dozbrojenie' systemu w dodatkowe czujniki itp.

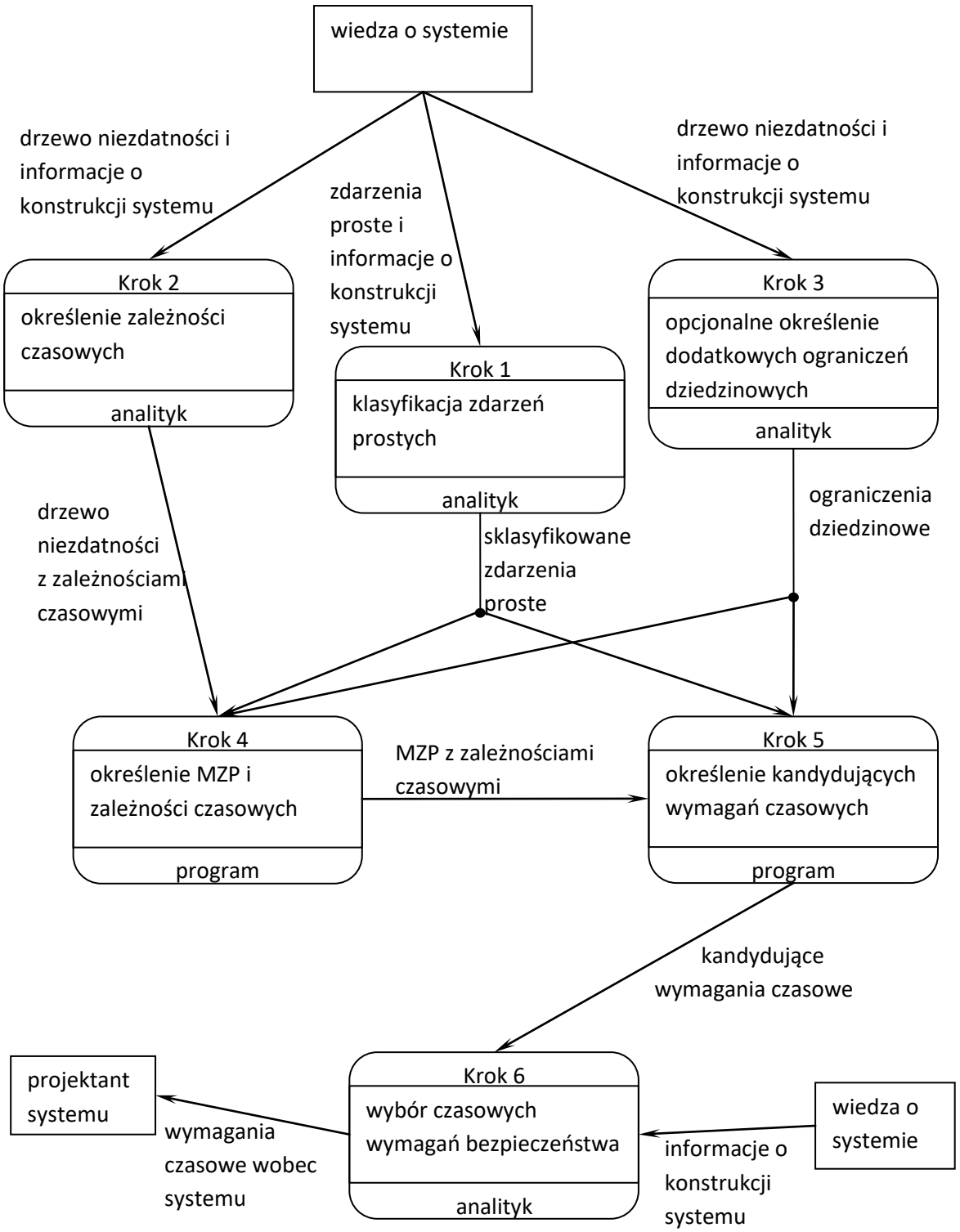
Przebieg metody obrazuje Rysunek 4.3. Przedstawiono na nim zależności pomiędzy poszczególnymi krokami algorytmu w postaci Diagramu Przepływu Danych (ang: Data Flow Diagram; DFD). Diagramy DFD zaproponowane zostały przez Larry'ego Constantine [SMC74, YC79] i nadal pozostają w użyciu [Amb04,LC09]. Diagram DFD przedstawia procesy przetwarzające dane (kwadraty z zaokrąglonymi krawędziami) oraz dane, które pomiędzy nimi przepływają (opisane strzałki). Dodatkowo można na nim umieścić magazyny danych (dwa równoległe odcinki) oraz terminatory (kwadraty; odbiorców bądź źródła danych znajdujące się poza modelowanym systemem).

Jak widać kroki 1-3 metody mogą być wykonane równolegle. Wejściem dla każdego z tych kroków jest tradycyjne drzewo niezdatności oraz informacje o konstrukcji rozpatrywanego systemu. W kroku 1, na podstawie drzewa niezdatności oraz znajomości budowy systemu, dla każdego zdarzenia prostego wyodrębnionego z drzewa niezdatności przypisywana jest odpowiednia kategoria (kontrolowalne, przewidywalne, obserwowalne lub nieobserwowalne). W kroku 2 dla każdej bramki występującej w drzewie niezdatności określane są zależności czasowe jakie muszą wystąpić na wejściach bramki, aby mogło wystąpić zdarzenie na wyjściu tej bramki. Określane są również zależności czasowe pomiędzy zdarzeniami wejściowymi a zdarzeniem wyjściowym poszczególnych bramek. W kroku 3 określane są zależności czasowe, które są inwariantne w rozpatrywanym systemie i wynikają ze specyfiki dziedziny aplikacyjnej (np. z praw fizyki, zewnętrznych procedur, itp.).

Produkty tych trzech kroków stanowią wejście do kroku 4, w którym określane są Minimalne Zbiory Przyczyn rozszerzone o czas. Krok ten jest wykonany automatycznie, co zobrazowano przez określenie „programu" jako wykonawcy kroku. W tym kroku dokonana wcześniej klasyfikacja zdarzeń służy ograniczaniu rozmiaru rozwiązania (a przez to i czasu przetwarzania) produkowanego przez ten krok. Krok 4 składa się z dwóch etapów. W pierwszym z nich wykonywany jest jeden z algorytmów: MZPCALC_FULL, MZPCALC_CONTROLABLE, MZPCALC_PAIR, MZPCALC_PART (patrz rozdział 6). W drugim etapie wykonywany jest algorytm REDUCE_MZP lub REDUCE_MZP_ext (patrz podrozdział 6.1).

W następnym kroku (kroku 5) odnajdywane są Minimalne Zbiory Przyczyn zawierające zdarzenia kontrolowalne i dla każdego z takich zbiorów sprawdzane jest, czy można sterować czasem wystąpienia zdarzenia kontrolowalnego tak, aby uniemożliwić spełnienie związanych z nim zależności czasowych. Zadanie to jest wykonywane przez algorytm REQCALC (podrozdział 7.1).

[8] Przez scenariusz zagrożenia rozumiany jest zestaw zależności czasowych opisujących w jakich warunkach wystąpienie akcji zadanych zdarzeń umożliwia materializację zagrożenia. Pojedynczemu MZP może być przypisany jeden bądź więcej scenariuszy zagrożeń.



Rysunek 4.3 Diagram Przepływu Danych metody TREM

Zidentyfikowane w Kroku 5 zależności czasowe porównywane są w z wyspecyfikowanymi w Kroku 3 ograniczeniami dziedzinowymi, aby odfiltrować te z nich, które są sprzeczne z tymi ograniczeniami. Pozostałe tworzą zbiór kandydujących wymagań czasowych. Krok 5 jest również wykonany

automatycznie. Krok 6, tak jak Kroki 1, 2 i 3, jest wykonywany przez człowieka. W tym kroku wymagania kandydujące są konfrontowane z wiedzą o systemie aby określić, które z nich są możliwe do spełnienia. Wymagania te tworzą listę wymagań czasowych, których spełnienie poprawi bezpieczeństwo systemu.

4.3 Przyjęte modele

4.3.1 Kategorie zdarzeń

W metodzie TREM modele statyczny i dynamiczny systemu poddawanego analizie przyjęto zgodnie z podrozdziałami 3.3 oraz 3.2 (odpowiednio). W modelu dynamicznym wprowadzono podział zdarzeń na kategorie. Określają one możliwość kontroli przez system występowania zdarzeń w nim zachodzących (w tym czasu ich wystąpienia) oraz „świadomości" systemu odnośnie występujących w nim zdarzeń. Jako typowy element systemu sprawujący funkcje kontrolno-obserwacyjne przyjęto wbudowany elektroniczny układ sterujący (choć nie wyklucza się innych rozwiązań).

Zdarzenia podzielono na:

- *kontrolowalne* – projektanci analizowanego systemu są w stanie kontrolować występowanie takich zdarzeń w systemie oraz czas ich wystąpienia, np. poprzez odpowiednie zaprogramowanie układu sterującego (sygnały wysyłane z układu sterującego),
- *przewidywalne* – wystąpienie zdarzenia w systemie da się przewidzieć (na podstawie wiedzy o wystąpieniach innych, poprzedzających je zdarzeń) przed jego materializacją,
- *obserwowalne* – wystąpienie zdarzenia daje się bezpośrednio zaobserwować (np. poprzez sygnały z czujników trafiające do układu sterującego),
- *nieobserwowalne* – wystąpienie zdarzenia w systemie lub jego środowisku nie zostanie bezpośrednio zaobserwowane (np. nie istnieją odpowiednie do tego celu czujniki, bądź ich użycia nie przewidziano w systemie) oraz nie jest możliwe do przewidzenia na podstawie zaobserwowanych zdarzeń.

Należy tu podkreślić, że powyższa klasyfikacja jest dokonywana z perspektywy systemu sterującego przy założeniu, że znane jest już drzewo niezdatności analizowanego systemu, a więc przy założeniu że dysponujemy dostateczną wiedzą na temat tego systemu. Dla przykładu, dodanie do systemu dodatkowego czujnika może spowodować, że zdarzenie nieobserwowalne stanie się obserwowalnym lub przewidywalnym.

Dodatkowo mogą wystąpić sytuacje graniczne, dla których klasyfikacja zdarzeń będzie zależała od analityka dokonującego klasyfikacji. Dla przykładu istnieje możliwość, że zdarzenie w systemie jest inicjowane przez system kontrolny, ale występuje duże bądź zmienne opóźnienie pomiędzy wysłaniem sygnału sterującego a nastąpieniem zdarzenia. W takiej sytuacji to analityk musi ocenić, czy stopień kontroli nad rozpatrywanym wydarzeniem jest wystarczający aby sklasyfikować je jako kontrolowalne, czy też należy je rozpatrywać jako przewidywalne.

Zakłada się również, że podane wyżej kategorie zdarzeń są rozłączne i przyjmuje się następujące priorytety poszczególnych kategorii: 1. Kontrolowalne, 2. Przewidywalne, 3. Obserwowalne, 4. Nieobserwowalne. Tak więc klasyfikując dane zdarzenie najpierw bierzemy pod uwagę jego przynależność do kategorii 1, jeżeli decyzja jest negatywna sprawdzamy jego przynależność do kategorii 2 itd.

Formalnie można to zapisać następująco:

(xviii) $E = C \cup U$, gdzie

E - zbiór zdarzeń mogących wystąpić w systemie

C - zbiór zdarzeń *kontrolowalnych*, oraz

U - zbiór zdarzeń *niekontrolowalnych*.

Zbiór U natomiast jest sumą trzech rozłącznych zbiorów:

(xix) $U = P \cup O \cup N$,

gdzie poszczególne symbole oznaczają:

P - zbiór zdarzeń *przewidywalnych*,

O - zbiór zdarzeń *obserwowalnych*,

N - zbiór zdarzeń *nieobserwowalnych*.

Przykład 7.

Jeżeli dla Przykład 4 przyjmiemy następujące oznaczenia zdarzeń:

- SZ - sygnał sterujący zapór drogowych,
- PP - pociąg na przejeździe,
- ZP - sygnał zbliżania się pociągu,
- SP - samochód na przejeździe,

to zgodnie z opisem systemu otrzymamy:

(xx) $SZ \in C$,

(xxi) $PP \in P$,

(xxii) $ZP \in O$,

(xxiii) $SP \in N$.

4.3.2 **Definicja drzewa niezdatności rozszerzonego o czas**

Niech *FT* oznacza zbiór drzew niezdatności, gdzie każde drzewo $ft \in FT$ jest zbiorem bramek:

(xxiv) $\forall_{ft \in FT} ft = \bigcup_{i=1}^{n} G_i$,

Każda bramka G_i jest charakteryzowana jednym zdarzeniem wyjściowym i zbiorem zdarzeń wejściowych i jest zdefiniowana za pomocą funkcji semantycznej M mającej postać ([BCG91, GW95]):

(xxv) $\mathrm{M}\left(\mathrm{G_j}\left(\mathrm{OUTPUT_j} \in \mathrm{E}, \mathrm{INPUT_j} \subseteq \mathrm{E}\right)\right) \equiv \forall_{o \in \phi(OUTPUT_j)} occur(o) \Longrightarrow \bigvee_{k=1}^{m} (\bigwedge_{l=1}^{n} (\exists i_{kl} \in \phi(I_{kl}) \cdot I_{kl} \in INPUT_j \wedge occur(i_{kl})) \wedge enabling\_condition_k(o, i_{k1}, \cdots, i_{kn}))$,

gdzie $enabling\_condition$ oznacza listę zależności czasowych pomiędzy zdarzeniami wejściowymi oraz wyjściowym.

Dodatkowo wymaga się, aby:

- zdarzenia wyjściowe różnych bramek były różne,
- każde zdarzenie wejściowe było przypisane dokładnie do jednej bramki.
- istniało tylko jedno zdarzenie szczytowe (zdarzenie wyjściowe pewnej bramki nie będące zdarzeniem wejściowym dla jakiejkolwiek bramki w danym drzewie).

Warunki te można wyrazić jako:

(xxvi) $\forall_{ft\in FT, O\in E} \cdot O = OUTPUT_l \Longrightarrow \forall_{i=1}^{n} O \neq OUTPUT_i \vee i = l,$

(xxvii) $\forall_{ft\in FT, I\in E} \cdot I \in INPUT_l \Longrightarrow \forall_{i=1}^{n} I \notin INPUT_i \vee i = l,$

(xxviii) $\forall_{ft\in FT, I\in E} \cdot \exists! \, k \cdot O = OUTPUT_k \wedge O \notin \bigcup_{j=1}^{n} INPUT_j,$

gdzie $OUTPUT_l$ to zdarzenie wyjściowe bramki G_l, a $INPUT_l$ to zbiór jej zdarzeń wejściowych.

Na podstawie analizy tak zdefiniowanego drzewa można otrzymać Minimalne Zbiory Przyczyn rozszerzone o warunki czasowe. Format Minimalnego Zbioru Przyczyn rozszerzonego o czas to:

(xxix) $\{e_1, e_2, \cdots, e_n | e_1 \in \phi(E_1) \wedge e_2 \in \phi(E_2) \wedge \cdots \wedge e_n \in \phi(E_n) \wedge enabling\_condition(e_1, e_2, \cdots, e_n)\},$

gdzie $enabling\_condition$ zawiera zależności czasowe pomiędzy akcjami zawartymi w argumentach, w szczególności dla każdej akcji zawartej w Minimalnym Zbiorze Przyczyn $enabling\_condition$ zawiera $occur(e_i)$.

Minimalność tak zdefiniowanych zbiorów przyczyn rozumiana jest standardowo. Usunięcie dowolnego zdarzenia z MZP powoduje niemożliwość spowodowana przez te przyczyny zagrożenia (niezależnie od warunków czasowych pomiędzy nimi).

Wyrażenia zawarte w $enabling\_condition$ opisują ograniczenia czasowe nałożone na wystąpienia zdarzeń z Minimalnych Zbiorów Przyczyn warunkujące możliwość wystąpienia zagrożenia. Innymi słowy wystąpienie wszystkich zdarzeń z Minimalnego Zbioru Przyczyn nie spełniające jednak warunków czasowych opisanych przez $enabling\_condition$ nie ma daje warunków dla zaistnienia zagrożenia.

5 Temporalna analiza drzew niezdatności w metodzie TREM

W rozdziale 3.5 przedstawiono algorytm TGRAF-MZP służący do analizy drzew niezdatności z czasem. Posługuje się on reprezentacją znormalizowanych wyrażeń czasowych w postaci T-GRAFu, który jest budowany na podstawie analizowanego drzewa niezdatności. T-GRAF jest skierowanym grafem o krawędziach opisanych liczbą rzeczywistą oraz relacją ('>' lub '≥') reprezentującymi relację czasową oraz wierzchołkami reprezentującymi tranzycje (początek bądź koniec wybranej akcji).

W metodzie TREM algorytm ten został rozszerzony w następujący sposób.

5.1 Analiza alternatywnych wyrażeń czasowych

Modyfikację tę oparto na spostrzeżeniu, że algorytm TGRAF-MZP przetwarza wyrażenia normalizowalne do postaci koniunkcji. Ogranicza to repertuar predykatów, które można użyć w analizie. Przykładowy predykat, który jest w ten sposób eliminowany z dostępności w analizie to:

$duration(e_1, e_2) < x$ (czas nakładania się akcji e_1 oraz e_2 jest mniejszy od x).

Zauważmy jednak, że ponieważ (z definicji Minimalnych Zbiorów Przyczyn) prawdziwa jest implikacja

$occur(zagrożenie) \Longrightarrow \bigvee_i MZP_i$ (przez $\bigvee_i MZP_i$ oznaczono alternatywę wszystkich Minimalnych Zbiorów Przyczyn zidentyfikowanych w danym drzewie niezdatności)

oznacza to, że poszczególne T-grafy tworzone w analizie TGRAF-MZP można traktować jako połączone operatorem alternatywy. Tak więc po normalizacji wyrażeń poszczególnych bramek wystarczy sprowadzić uzyskane wyrażenie do normalnej postaci sumacyjnej [AP95] (ang. disjunctive normal form, DNF; alternatywa wyrażeń, z których każde to koniunkcja literałów) i przedstawić każdy iloczyn logiczny obecny w wynikowym wyrażeniu jako osobny T-graf, aby możliwe było stosowanie operatora alternatywy w wyrażeniach czasowych.

Postępowanie takie może sprawić, że przy poprawnej definicji wyrażeń czasowych w bramkach, po normalizacji tych definicji i sprowadzeniu wyniku do normalnej postaci sumacyjnej niektóre alternatywy będą sprzeczne w sensie logicznym (wyrażenie $x < y \wedge y < x$ jest nieprawdziwe niezależnie od wartościowania wolnych zmiennych x i y). Takie wewnętrznie sprzeczne alternatywy są wyłączane poza zakres dalszych analiz.

Przykład 8.

Rozważmy wyrażenie $duration(e_1, e_2) < t \wedge start(e_1) < start(e_2)$. Po normalizacji $duration(e_1, e_2) < t$ otrzymamy:

(xxx) $start(e_1) \leq start(e_2) \wedge start(e_2) < end(e_2) \wedge end(e_2) \leq end(e_1) \wedge start(e_2) + t > end(e_2)$
$\vee\ start(e_2) \leq start(e_1) \wedge start(e_1) < end(e_1) \wedge end(e_1) \leq end(e_2) \wedge start(e_1) + t > end(e_1)$
$\vee\ start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge start(e_1) \leq start(e_2) \wedge start(e_2) \leq end(e_1) \wedge end(e_1) \leq end(e_2) \wedge start(e_2) + t > end(e_1)$
$\vee\ start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge start(e_2) \leq start(e_1) \wedge start(e_1) \leq end(e_2) \wedge end(e_2) \leq end(e_1) \wedge start(e_1) + t > end(e_2)$

Powyższe zachowania systemu zostały przedstawione na Rysunek 5.1.

Co po dodaniu warunku $start(e_1) < start(e_2)$ i sprowadzeniu do normalnej postaci sumacyjnej daje:

(xxxi) $start(e_1) \leq start(e_2) \wedge start(e_2) < end(e_2) \wedge end(e_2) \leq end(e_1) \wedge start(e_2) + t > end(e_2) \wedge start(e_1) < start(e_2) \vee$

(xxxii) $\boldsymbol{start(e_2) \leq start(e_1)} \wedge start(e_1) < end(e_1) \wedge end(e_1) \leq end(e_2) \wedge start(e_1) + t > end(e_1) \wedge \boldsymbol{start(e_1) < start(e_2)} \vee$

(xxxiii) $start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge start(e_1) \leq start(e_2) \wedge start(e_2) \leq end(e_1) \wedge end(e_1) \leq end(e_2) \wedge start(e_2) + t > end(e_1) \wedge start(e_1) < start(e_2) \vee$

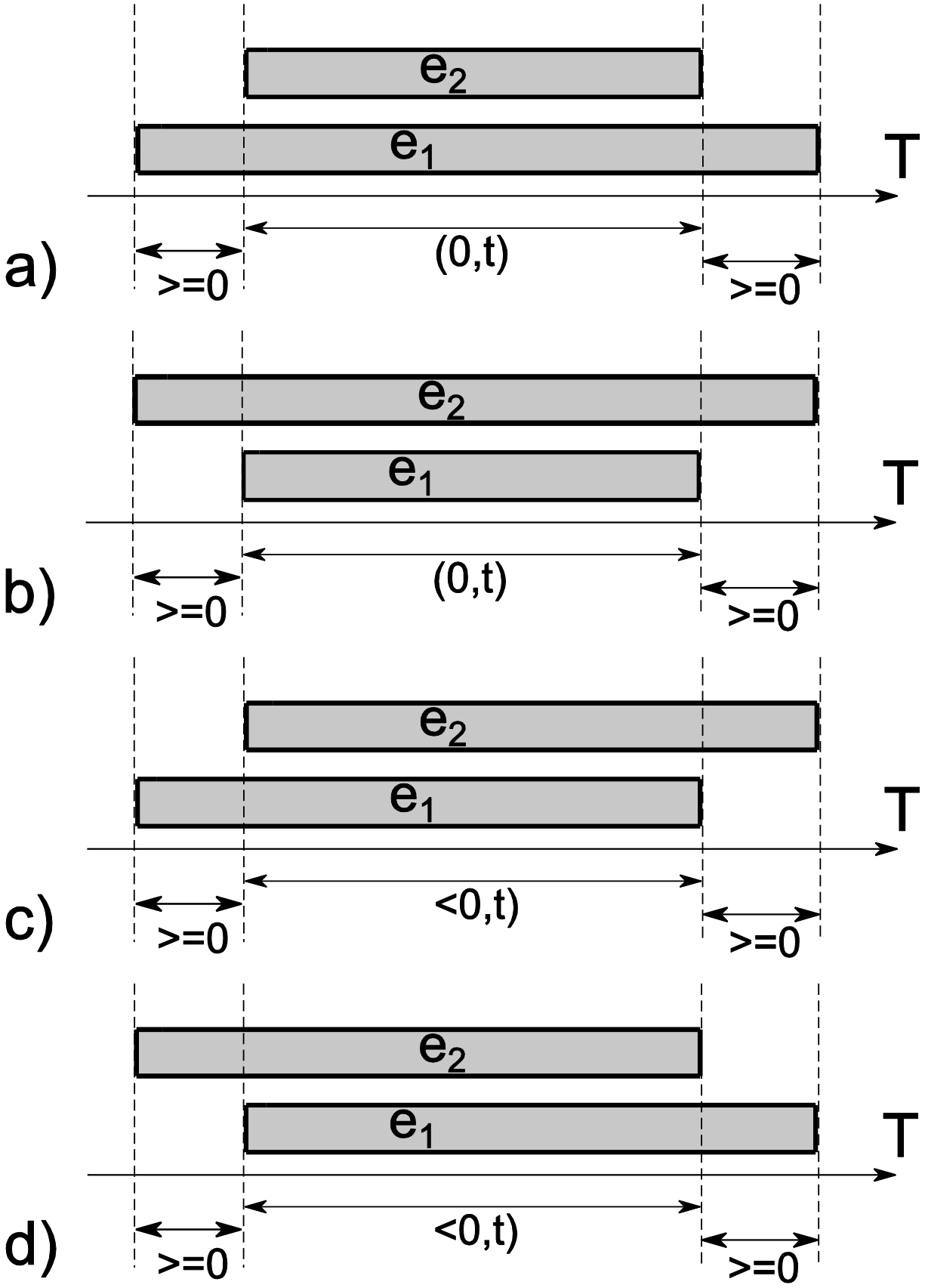
(xxxiv) $start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge \boldsymbol{start(e_2) \leq start(e_1)} \wedge start(e_1) \leq end(e_2) \wedge end(e_2) \leq end(e_1) \wedge start(e_1) + t > end(e_2) \wedge \boldsymbol{start(e_1) < start(e_2)}$

Jak można zauważyć alternatywy (xxxii) oraz (xxxiv) są wewnętrznie sprzeczne[9] (dla ułatwienia sprzeczne ze sobą wyrażenia zostały umieszczone na szarym tle).

Po odrzuceniu sprzecznych alternatyw (jako wyrażenia o logicznej wartości *fałsz* nie wpływają na wartościowanie całości wyrażenia) otrzymujemy:

(xxxv) $start(e_1) \leq start(e_2) \wedge start(e_2) < end(e_2) \wedge end(e_2) \leq end(e_1) \wedge start(e_2) + t > end(e_2) \wedge start(e_1) < start(e_2) \vee$
$start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge start(e_1) \leq start(e_2) \wedge start(e_2) \leq end(e_1) \wedge end(e_1) \leq end(e_2) \wedge start(e_2) + t > end(e_1) \wedge start(e_1) < start(e_2)$

[9] Sprzeczność można wykryć w reprezentacji T-grafowej wyrażeń czasowych poprzez znalezienie cyklu ujemnego lub cyklu o wadze $(0, >)$ (co odpowiadałoby odpowiednio wyrażeniom $x - t \geq x$ dla $(t > 0)$ oraz $x > x$).



Rysunek 5.1 Możliwe relacje akcji e_1 i e_2 wyznaczone wyrażeniem duration(e_1,e_2)<t

Sytuacja występowania wewnętrznie sprzecznych alternatyw jest w pełni akceptowalna. Alternatywy takie w procesie analizy są odrzucane, co prowadzi do uproszczenia wyrażenia poddawanego dalszej analizie. Jako nieakceptowalną traktuje się jedynie sytuację, w której wszystkie alternatywy są wewnętrznie sprzeczne.

Uogólniając, po przetworzeniu wyrażeń czasowych opisujących konkretną bramkę na odpowiadające jej T-grafy, otrzymane T-grafy sprawdzane są pod kątem występowania sprzeczności. Sprzeczne T-grafy są odrzucane (nie biorą udziału w dalszej analizie). Jako błąd podważający zasadność analizowania drzewa (błędna definicja bramki) traktuje się niewystąpienie choć jednego niesprzecznego T-grafu w grafowej reprezentacji jakiejkolwiek bramki.

Przykład 9.

> W powyższym przykładzie w przypadku wyrażeń (xxxii) i (xxxiv) otrzymane dla nich T-grafy są wewnętrznie sprzeczne ponieważ występuje w nich dwukrawędziowy cykl $\left(start(e_1), start(e_2), start(e_1)\right)$ lub $\left(start(e_2), start(e_1), start(e_2)\right)$ (w zależności od przyjętego wierzchołka początkowego) o wadze $(0, >)$ i oznaczający odpowiednio $start(e_1) < start(e_1)$ oraz $start(e_2) < start(e_2)$.

Takie podejście zmienia również sposób w jaki łączone są definicje bramek w celu analizy wyrażeń czasowych w poszczególnych MZP. W sensie logicznym bramki łączone są poprzez mnożenie ich wyrażeń czasowych (połączenie logicznym operatorem iloczynu), a otrzymane wyrażenie należy ponownie sprowadzić do normalnej postaci sumacyjnej. W sensie algorytmicznym T-grafy reprezentujące poszczególne bramki łączymy na zasadzie każdy-z-każdym poprzez uwspólnienie wierzchołków o tych samych etykietach. Bardziej formalnie można to zapisać jako

(xxxvi) $\bigcup_{a\in A, b\in B}(a \dot{+} b)$,

gdzie A i B reprezentują zbiory T-grafów dwóch łączonych bramek, a

(xxxvii) $\dot{+}: T-graf \times T-graf \rightarrow T-graf$

to operacja tworząca nowy T-graf z zadanych argumentów taki, że jego zbiór wierzchołków to suma zbiorów wierzchołków grafów będących argumentami, a zbiór jego krawędzi to suma zbiorów krawędzi grafów będących argumentami, z zastrzeżeniem, że jeśli pomiędzy dwoma wierzchołkami w tę samą stronę biegną dwie krawędzie, to w wynikowym T-grafie umieszczana jest tylko krawędź o niższej wadze.

Takie podejście umożliwia używanie w definicji zależności czasowych w drzewie niezdatności m.in. następujących predykatów (wszelkie definicje dla $t \geq 0$):

Tabela 5.1 Przykładowe wyrażenia ECSDM, których normalizacja wymaga użycia operatora alternatywy

Wyrażenie ECSDM	Odpowiednik w postaci normalnej ECDSM
$duration(e_1, e_2) < t$	$start(e_1) \leq start(e_2) \wedge start(e_2) < end(e_2) \wedge$ $end(e_2) \leq end(e_1) \wedge start(e_2) + t > end(e_2) \vee$ $start(e_2) \leq start(e_1) \wedge start(e_1) < end(e_1) \wedge$ $end(e_1) \leq end(e_2) \wedge start(e_1) + t > end(e_1) \vee$ $start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge$ $start(e_1) \leq start(e_2) \wedge start(e_2) \leq end(e_1) \wedge$ $end(e_1) \leq end(e_2) \wedge start(e_2) + t > end(e_1) \vee$ $start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge$ $start(e_2) \leq start(e_1) \wedge start(e_1) \leq end(e_2) \wedge$ $end(e_2) \leq end(e_1) \wedge start(e_1) + t > end(e_2)$
$cduration(e_1, e_2) > t$	$start(e_1) \leq start(e_2) \wedge start(e_2) < end(e_2) \wedge$ $end(e_2) \leq end(e_1) \wedge start(e_1) + t < end(e_1) \vee$ $start(e_2) \leq start(e_1) \wedge start(e_1) < end(e_1) \wedge$ $end(e_1) \leq end(e_2) \wedge start(e_2) + t < end(e_2) \vee$ $start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge$ $start(e_1) \leq start(e_2) \wedge start(e_2) \leq end(e_1) \wedge$ $end(e_1) \leq end(e_2) \wedge start(e_1) + t < end(e_2) \vee$ $start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge$ $start(e_2) \leq start(e_1) \wedge start(e_1) \leq end(e_2) \wedge$ $end(e_2) \leq end(e_1) \wedge start(e_2) + t < end(e_1)$
$any\_cause(e_1, e_2, e_3)$	$start(e_1) < end(e_1) \wedge start(e_2) < end(e_2) \wedge$ $start(e_2) < start(e_1) \wedge end(e_2) < end(e_1) \vee$ $start(e_1) < end(e_1) \wedge start(e_3) < end(e_3) \wedge$ $start(e_3) < start(e_1) \wedge end(e_3) < end(e_1)$

Pełna lista możliwych do wykorzystania wyrażeń ECSDM znajduje się w raporcie [Gol17b]. Lista ta zawiera 22 pozycje (70 licząc wszystkie kombinacje funkcji min/max). Bez zastosowania alternatywy w normalizacji wydarzeń możliwe jest zdefiniowanie jedynie 14 z nich.

Jak widać na powyższym przykładzie podejście to niesie ze sobą zagrożenie eksplozji kombinatorycznej, gdyż normalizacja predykatów dwuargumentowych może skutkować powstaniem nawet czterech alternatyw. Mając dwa tego typu wyrażenia w bramce otrzymujemy szesnaście alternatyw. Podobnie ma się sytuacja przy łączeniu wyrażeń dla poszczególnych bramek (za każdym razem musi być utrzymana normalna postać sumacyjna). Niemniej sytuacja taka będzie występowała niejako wyłącznie „na życzenie" analityka (poprzez użycie wyrażeń wymagających użycia alternatywy w trakcie normalizacji) i nie powoduje dodatkowych narzutów dla wyrażeń normalizujących się do postaci iloczynu.

5.2 Obliczanie rozwiązań częściowych

Drugie usprawnienie algorytmu TGRAF-MZP wynika z obserwacji, że algorytm PTGRAF (używany przez algorytm TGRAF-MZP) jest w istocie algorytmem wyszukiwania wszystkich najlżejszych[10] ścieżek w cyklicznym grafie skierowanym o funkcji wagowej

(xxxviii) $w: F \to \{\mathbb{R} \times \{>, \geq\}\}$,

gdzie F oznacza zbiór wszystkich krawędzi, a $\mathbb{R}$ to zbiór liczb rzeczywistych. Można zauważyć, że w grafie cyklicznym nie da się określić najlżejszych ścieżek jeśli cykle mogą mieć wagi ujemne (niezależnie od przyjętego algorytmu). A dokładniej, dla przyjętej powyżej funkcji wagowej, waga wszystkich cykli musi być dodatnia lub wynosić $(0, \geq)$. Nie stanowi to ograniczenia, gdyż powyższe ograniczenie występuje również w dziedzinie zależności czasowych. Jak zaznaczono powyżej, wystąpienie w grafie cyklu ujemnego bądź $(0, >)$ stanowi sprzeczność – tranzycja musi wystąpić przed swoim własnym wystąpieniem. I choć żaden algorytm nie jest wtedy w stanie obliczyć najlżejszych ścieżek, jest możliwe wykrycie w trakcie pracy algorytmu cykli ujemnych i odrzucenie takich grafów z dalszej analizy.

Ponadto można zauważyć, że przy przetwarzaniu osobno T-grafu dla każdego MZP pewne bramki będą przetwarzane wielokrotnie (im bliżej szczytu drzewa, tym częściej). To prowadzi do pytania, czy da się wydajniej obliczyć $enabling\_condition$ dla MZP poprzez użycie innego algorytmu obliczania najlżejszych ścieżek (w szczególności umożliwiającego obliczanie najlżejszych ścieżek dla wybranego podzbioru wierzchołków) lub innej strategii budowania T-grafów do analizy. Dla potrzeb tej pracy przeprowadzono opisaną poniżej analizę mającą wyłonić najlepszą wersję algorytmu oraz strategii jego użycia.

5.2.1 Algorytmy obliczania najlżejszych ścieżek w grafie

Analizie poddano różne algorytmy określania najkrótszych[11] ścieżek w grafach.

Jednym z najpopularniejszych algorytmów dla określania wag najkrótszych ścieżek o jednym źródle jest algorytm Dijkstry, zaproponowany w [Dij59]. Nadaje się on jednak tylko do grafów o krawędziach

[10] O ile w literaturze wartość przypisaną krawędzi często określa się mianem wagi, to algorytmy wyszukiwania ścieżek o najmniejszej łącznej wadze określa się algorytmami znajdowania najkrótszych ścieżek. W tej pracy używany będzie jednak termin najlżejszych w celu uniknięcia skojarzeń z długością krawędzi (długość jest częścią wagi).

[11] W oryginale wymieniane algorytmy odnoszą się do krawędzi opisanych pojedynczą liczbą, stąd też inna nomenklatura. Zastosowanie ich w przedstawianej pracy będzie wymagać dopasowania ich do innej funkcji wagowej.

o wagach nieujemnych, nie może więc być bezpośrednio zastosowany w przypadku zależności czasowych.

Innym algorytmem dla problemu najkrótszych ścieżek z jednym źródłem jest algorytm Bellmana-Forda [CLRS05], który może zostać zastosowany do grafów o ujemnych krawędziach, ale ma wyższą niż algorytm Dijkstry złożoność obliczeniową. Jednym z kroków w tym algorytmie jest sprawdzenie, czy w grafie nie istnieją cykle o wadze ujemnej (osiągalne ze źródła). W rozważanym w niniejszej pracy zastosowaniu test ten musiałby zostać uzupełniony o wykrywanie cykli o wadze $(0, >)$ ze względu na interpretację nadaną grafom.

Obydwa powyższe algorytmy mogą zostać wykorzystane do określenia długości najkrótszych ścieżek pomiędzy wszystkimi wierzchołkami w grafie poprzez zastosowanie ich po kolei do wszystkich wierzchołków. Interesującą ich kombinacją jest algorytm Johnsona[Joh77], który najpierw jednokrotnie wykorzystuje algorytm Bellmana-Forda do wykrycia cykli ujemnych i przypisania wszystkim wierzchołkom określonych wartości, które służą potem do stworzenia nowej funkcji wagowej *w'* (o wartościach nieujemnych), co umożliwia zastosowanie algorytmu Dijkstry dla wszystkich wierzchołków po kolei. Dla funkcji wagowej (xxxviii) problemem jest jednak powrót z funkcji *w'* do *w* (brak operacji odejmowania dla operatorów porównania). Rozwiązaniem może być użycie funkcji *w''* podczas przypisywania wierzchołkom wartości. *w''* można określić następująco:

(xxxix) $\forall_{f \in F} w(f) = (r, b) \Longrightarrow w''(f) = r$

Istnieją również algorytmy przeznaczone specjalnie do obliczania długości najkrótszych ścieżek pomiędzy wszystkimi wierzchołkami, m.in. algorytm Floyda-Warshala ([Flo62, War62]), który operuje na macierzowej reprezentacji grafu, lub algorytm Carre ([Car69, Car71]). Te dwa algorytmy są równoważne ([WJS05]).

Opisane algorytmy są podstawowymi i były przez lata udoskonalane. Często odbywało się to jednak kosztem ograniczenia zakresu wag do liczb całkowitych, bądź też rzeczywistych nieujemnych [PR05], albo poprzez ograniczenie grafów do grafów nieskierowanych [PR05]. Powstają też algorytmy dające gwarancje probabilistyczne, lecz nakładające również wymagania odnośnie rozkładu prawdopodobieństwa wartości wag krawędzi [Hag06]. Stworzono również algorytmy dla problemów pośrednich pomiędzy obliczaniem długości najkrótszych ścieżek dla jednego źródła i dla wszystkich par wierzchołków [WJS05].

5.2.2 Strategie określania zależności czasowych

Wyróżnienie innych niż opisana w [GW97] strategii (określanej dalej mianem *podstawowej*) budowania T-grafów do analizy opiera się na spostrzeżeniu, że graf stworzony według strategii podstawowej będzie zawierał warstwy – zbiory wierzchołków odpowiadające bramkom z tego samego poziomu drzewa. Wierzchołki na poszczególnych warstwach są połączone między sobą (w ramach pojedynczej bramki), oraz z maksymalnie dwoma wierzchołkami z warstwy wyższej. Są także połączone z pewną liczbą wierzchołków warstwy o jeden niższej. To powoduje, że obliczanie zależności czasowych pomiędzy zdarzeniami z Minimalnego Zbioru Przyczyn można prowadzić warstwowo – dla każdej kolejnej iteracji na wejściu podawać długości najlżejszych ścieżek pomiędzy wierzchołkami warstwy o jeden wyższej oraz zależności pomiędzy tą warstwą a rozpatrywaną, a jako

wyjście otrzymywać długości najlżejszych ścieżek dla danej warstwy. Powtarzając tę czynność aż do dołu grafu uzyska się zależności czasowe pomiędzy zdarzeniami z Minimalnego Zbioru Przyczyn. Podejście to ma tę zaletę, że obliczenia dla warstw wyższych można wykorzystać dla wielu minimalnych zbiorów przyczyn (podejście to będzie nazywane strategią *warstwową*). Można również zastosować jeszcze bardziej radykalne rozwiązanie, a mianowicie:

- do dotychczasowego grafu rozwiązania (na starcie dwa wierzchołki reprezentujące zdarzenie szczytowe) dodać graf reprezentujący bramkę, której zdarzenie wyjściowe jest już reprezentowane w grafie rozwiązania,
- znaleźć wagi najlżejszych ścieżek pomiędzy wszystkimi wierzchołkami grafu rozwiązania i dodać krawędź o obliczonej wadze pomiędzy wierzchołkami określającymi początek i koniec ścieżki (o ile już nie znajduje się w grafie),
- usunąć wierzchołki reprezentujące zdarzenie wyjściowe ostatnio dodanej bramki,
- powtarzać czynności aż do dodania wszystkich bramek.

Kolejność dodawania poszczególnych bramek do grafu rozwiązania też można zmieniać, na przykład zgodnie z metodą przeglądania drzewa wgłąb (strategia *wgłąb*) lub wszerz (strategia *wszerz*) (Dodawanie pojedynczych bramek jako strategię budowania rozwiązania przyjęto w [Jar96]).

Można wykazać następujące twierdzenie:

> ***Strategie bazujące na budowaniu rozwiązań częściowych, powadzą do wyliczenia właściwej wagi najlżejszych ścieżek pomiędzy zdarzeniami z każdego z Minimalnych Zbiorów Przyczyn.***

Dowód tego twierdzenia można oprzeć na twierdzeniu podanym w [Kor78], które można wyrazić następująco:

> ***Jeśli w grafie o krawędziach opisanych parą $\{R \times \{>, \geq\}\}$ waga każdej cyklicznej drogi prostej jest nieujemna, to każdy odcinek dowolnej drogi najlżejszej jest drogą najlżejszą.***

Twierdzenie to jest określone dla grafów o krawędziach opisanych liczbami rzeczywistymi (poza tym jednobrzmiące z dokładnością do przyjętej nomenklatury). Dowód twierdzenia (również zawarty w [Kor78]) jest jednak na tyle ogólny, że zamiana w nim $l: F \to R$ na $w: F \to \{R \times \{>, \geq\}\}$ zachowuje jego poprawność.

Twierdzenie to, przy określaniu najlżejszej ścieżki, umożliwia posłużenie się wcześniej obliczonymi najlżejszymi ścieżkami pomiędzy niektórymi wierzchołkami zamiast odwoływania się do wszystkich ścieżek istniejących pomiędzy tymi elementami.

5.2.3 Ocena złożoności algorytmów kandydujących

Do oceny poszczególnych algorytmów i strategii posłużono się kryterium ograniczenia górnego czasu wykonywania algorytmu przy optymalnej strategii jego zastosowania. Ponieważ nie dysponowano implementacjami wszystkich tych algorytmów, czas wykonania oszacowano przy użyciu

pesymistycznej złożoności obliczeniowej. Przyjęto, iż dla każdego z rozpatrywanych algorytmów stała ukryta w notacji $O(\cdot)$ jest zbliżona (dla algorytmów określania wagi najlżejszej ścieżki złożoność determinują te same operacje: porównanie wag dwóch krawędzi oraz sumowanie wag dwóch krawędzi).

Eksperyment polegał na ustaleniu, dla każdego z drzew testowych, dla ilu, oraz jak dużych (liczba wierzchołków i krawędzi) grafów trzeba będzie policzyć długości najlżejszych ścieżek dla każdej z rozpatrywanych strategii. Po czym oszacowano czas obliczania zależności czasowych poprzez podstawienie tych wartości do wzorów na złożoność obliczeniową (przyjmując dla wszystkich algorytmów współczynnik 1).

Rozpatrywane algorytmy:

- Belmana-Forda –$O(V^2E)$,
- Johnsona - $O(V(V+E)lg(V)+VE)$ - (zgodnie z wynikami przedstawionymi w [YH06] założono użycie w algorytmie Dijkstry kopca binarnego),
- Floyda-Warshala –$O(V^3)$.

We wszystkich powyższych wzorach V oznacza liczbę wierzchołków w przetwarzanym grafie, a E liczbę krawędzi.

Rozpatrywane strategie:

- strategia *podstawowa*,
- strategia *podstawowa – wybiórcza* (graf tworzony tak jak w strategii podstawowej, jednak długości najlżejszych ścieżek są obliczane tylko pomiędzy zdarzeniami z minimalnych zbiorów przyczyn),
- strategia *warstwowa*,
- strategia *wszerz*,
- strategia *wgłąb*.

Obliczenia przeprowadzono dla pięciu drzew niezdatności. Dwa z nich zostały skonstruowane na zasadzie najgorszego możliwego przypadku ([Skr05]) – składały się na przemian z dwuwejściowych bramek *AND* i *OR*, poczynając od bramki *AND* (ta miała na wejściach dwie bramki *OR*, każda z tych bramek miała na wejściach dwie bramki *AND*, itd.). Drzewo *T1* zawierało 15 bramek, drzewo *T2* zawierało 63 bramki. Ponadto stworzono drzewo *T3*, podobne do *T1* (też 15 bramek), lecz ze szczytową bramką *OR*. Dla tych drzew przyjęto maksymalną liczbę krawędzi w grafach poszczególnych bramek.

W związku z opinią zawartą w [Skr05], że takie drzewa niezdatności są w rzeczywistości mało prawdopodobne, dokonano również obliczeń dla dwóch rzeczywistych drzew niezdatności: *PG* – drzewa niezdatności dla systemu palnika gazowego (patrz podrozdział 9.1.2.1), oraz *PK* – drzewa niezdatności dla przejazdu kolejowego (podrozdział 9.1.2.2).

Zestawienie drzew niezdatności biorących udział w eksperymencie wraz z podstawowymi informacjami na ich temat zebrano w Tabela 5.2. Podaje ona oznaczenie drzewa, jego pochodzenie oraz parametry takie jak łączna liczba bramek oraz liczby bramek AND oraz OR oraz wysokość drzewa

(rozumiana jako maksymalna liczba bramek na ścieżce od korzenia do liścia). Należy również zaznaczyć, że choć w drzewach niezdatności poddanych analizie każda bramka ma dwa wejścia, to nie jest to wymagane przez proponowaną metodę. Maksymalna liczba wejść nie jest określona, natomiast minimalna to 1. Stąd łączna liczba bramek w drzewach GB i TC nie jest równa sumie bramek AND i OR (bramki o jednym wejściu nie są zaliczane do tych kategorii).

Tabela 5.2 Drzewa niezdatności biorące udział w eksperymencie

Oznaczenie	Źródło	Liczba bramek	Liczba bramek AND	Liczba bramek OR	Wysokość drzewa
PG	Skonstruowane dla systemu opisanego w [GW97]	7	1	5	4
PK	Skonstruowane dla systemu opisanego w [GG06]	10	2	5	4
T1	Skonstruowane sztucznie ([Skr05])	15	5	10	4
T2	Skonstruowane sztucznie ([Skr05])	63	21	42	6
T3	Skonstruowane sztucznie ([Skr05])	15	10	5	4

Liczbowe wyniki eksperymentu zgromadzone zostały w tabeli Tabela 5.3. Liczby w poszczególnych komórkach tabeli oznaczają łączną liczbę operacji potrzebnych do określenia zależności czasowych we wszystkich Minimalnych Zbiorach Przyczyn rozpatrywanego drzewa. Kolorem szarym w tabeli wyróżniono algorytm, który dla danej strategii budowania rozwiązania wymaga najmniej operacji. Natomiast kolorem niebieskim wyróżniono strategię, która dla zadanego drzewa wymagała najmniej operacji.

Z zaprezentowanych danych można odczytać, że najlepszymi strategiami obliczania zależności czasowych pomiędzy zdarzeniami z Minimalnych Zbiorów Przyczyn są strategie polegająca na budowaniu rozwiązania poprzez powiększanie rozwiązań cząstkowych o jedną bramkę naraz (strategie *wszerz* i *wgłąb*).

Spośród nich strategia *wszerz* wydaje się być bardziej skuteczna – okazała się lepsza od konkurentki w czterech przypadkach na pięć (biorąc pod uwagę najlepszy z algorytmów), nie jest to jednak przewaga duża, a jedyna przegrana nastąpiła dla rzeczywistego drzewa niezdatności.

Wśród algorytmów najlepszy okazał się algorytm Floyda-Warshala, który wygrał dla wszystkich drzew niezdatności w trzech z czterech strategii, w których można go było zastosować. Przegrał (z wyjątkiem dwóch drzew niezdatności) dla strategii *podstawowej*. Jest to strategia zakładająca przetwarzanie dużych, względnie rzadkich grafów, stąd lepiej wypadł tu algorytm Johnsona. Przypadki wygranej algorytmu Floyda-Warshala w strategii *podstawowej* zdarzyły się dla małych konstruowanych drzew niezdatności (mały rozmiar drzewa, gęstsze od przypadków rzeczywistych).

Można się zastanawiać, czy określenie innego sposobu szeregowania bramek nie umożliwi uzyskania strategii lepszej niż strategia *wszerz* i *wgłąb*, jednak różnice między tymi strategiami były tak niewielkie, że tę kwestię pozostawiono nie zbadaną.

Na podstawie otrzymanych wyników wybrano algorytm Floyda-Warshala oraz strategię *wszerz* do implementacji w narzędziu analitycznym. Powyższe rozważania przedstawione zostały również w [Gol07b].

Tabela 5.3 Wyniki eksperymentu

Strategia	Algorytm	Drzewo				
		GB	*TC*	*T1*	*T2*	*T3*
Wszerz	Bellman-Ford	10800	446064	831272	2653901608	78928
	Johnson	9636	305544	403585	806976641	43122
	Floyd-Warshal	7344	219488	127560	191405640	15248
Wgłąb	Bellman-Ford	11232	363568	818664	2649042968	82832
	Johnson	9988	246056	402753	818528901	44474
	Floyd-Warshal	7560	170848	128008	199570296	15568
Warstwowa	Bellman-Ford	18088	540228	1618744	5376518552	97264
	Johnson	13856	337084	578411	1092706375	45622
	Floyd-Warshal	11216	260048	271320	538025784	18480
Podstawowa	Bellman-Ford	188392	1931040	6316544	18629132288	358400
	Johnson	82454	857352	1588288	2525609984	109200
	Floyd-Warshal	99344	1306368	1124864	3196715008	64000
Podstawowa - wybiórcza	Bellman-Ford	46160	643680	971776	2569535488	71680
	Johnson	29084	357304	449920	625262592	36176
	Floyd-Warshal	NA	NA	NA	NA	NA

6 Algorytm MZPCALC_FULL obliczania minimalnych zbiorów przyczyn

Prezentowany algorytm, nazwany MZPCALC_FULL, oblicza MZP poprzez dodawanie do wyniku cząstkowego pojedynczych bramek według strategii *wszerz*, wyznaczanie najlżejszych ścieżek zgodnie z algorytmem Floyda-Warshala oraz odrzucanie zdarzeń pośrednich.

Algorytm MZPCALC_FULL składa się z następujących kroków:

ALGORYTM

MZPCALC_FULL

KROKI

1. Znormalizuj wyrażenia bramki szczytowej drzewa niezdatności zgodnie z [Gol17b] (patrz Tabela 5.1), sprowadź je do normalnej postaci sumacyjnej (DNF) a każdy iloczyn przedstaw w postaci T-grafu.
2. Oblicz najlżejsze ścieżki we wszystkich z utworzonych grafów po czym usuń z nich tranzycje zdarzenia szczytowego. Jeśli nie uzyskano żadnego T-grafu przerwij algorytm.
3. Do pustej listy FIFO dodaj wszystkie bramki znajdujące się na wejściu bramki szczytowej (bramki, których zdarzeniami wyjściowymi są zdarzenia wejściowe bramki szczytowej).
4. Dopóki lista nie będzie pusta powtarzaj:
    a. Pobierz pierwszą bramkę z listy bramek,
    b. Do listy bramek dodaj wszystkie bramki będące na wejściu pobranej bramki,
    c. Znormalizuj wyrażenia czasowe tej bramki, sprowadź do DNF i przedstaw w postaci T-grafów. Oblicz najlżejsze ścieżki. Jeśli nie uzyskano żadnego T-grafu, przerwij algorytm.
    d. Oblicz produkt kartezjański dotychczasowych T-grafów i T-grafów uzyskanych w poprzednim punkcie.
    e. Każdą parę T-grafów, które zawierają wspólne zdarzenie, połącz w jeden T-graf. Oblicz w nim najlżejsze ścieżki po czym usuń zdarzenie wspólne dla obu tworzących T-grafów.
    f. Dla par T-grafów, które nie miały wspólnych zdarzeń pozostaw dotychczasowy T-graf (a nie graf z punktu 4.c).
5. Z powstałej listy usuń T-grafy nadmiarowe (patrz algorytm REDUCE_MZP w rozdziale 6.1).

Bardziej precyzyjny (oparty o język formalny) opis powyższego algorytmu, jak i pozostałych algorytmów przedstawionych w pracy znajduje się w raporcie [Gol17c].

W algorytmie MZPCALC_FULL przyjmowane jest dodatkowe założenie odnośnie konstrukcji bramek w drzewie niezdatności (dodatkowe w stosunku do modelu przedstawionego w podrozdziale 3.4).

Założenie: w bramkach OR łączne wystąpienie obydwu zdarzeń wejściowych nie jest brane pod uwagę.

Jest to założenie zbieżne z założeniem przyjmowanym przy probabilistycznej analizie drzew niezdatności, gdzie przy wyliczaniu prawdopodobieństwa wystąpienia zagrożenia nie bierze się pod uwagę prawdopodobieństwa łącznego wystąpienia zdarzeń w bramkach *OR*.

W analizie probabilistycznej argumentuje się to oczekiwanymi niskimi prawdopodobieństwami występowania zdarzeń i ich zakładaną niezależnością, co powoduje, że prawdopodobieństwo łącznego wystąpienia takich zdarzeń byłoby o kilka rzędów wielkości mniejsze [NUREG0492].

W analizie czasowej nie można przyjąć tak prostego założenia. Można jednak oczekiwać, iż zajdzie jeden z przypadków:

1. wystąpienie łączne obu zdarzeń nie będzie się przekładało na zwiększenie możliwości wystąpienia zdarzenia szczytowego względem wystąpienia tylko jednego ze zdarzeń (w istocie będzie więc oznaczało, że bramka reprezentuje alternatywę wykluczającą), albo
2. wpływ łącznego wystąpienia tych zdarzeń na zdarzenie szczytowe został jawnie zamodelowany w drzewie jako bramka *AND* ($A\ AND\ B\ \equiv (A\ XOR\ B) XOR (A'\ AND\ B')$).

Przykład 10.

Wróćmy na chwilę do przypadku 1. Dla ilustracji przyjmijmy, że mamy bramkę zdefiniowaną jako:

(xl) $\mathrm{M}\big(OR(A,\{B,C\})\big) \equiv \forall_{a\in\phi(A)} occur(a) \Longrightarrow \exists b \in$
$\phi(\mathrm{B}) \cdot \mathrm{occur(b)} \wedge \mathrm{enabling\_condition_B(a,b)}$
$\vee\ \exists c \in \phi(\mathrm{C}) \cdot \mathrm{occur(c)} \wedge \mathrm{enabling\_condition_C(a,c)}$
$\vee\ \exists b \in \phi(\mathrm{B}), \mathrm{c} \in \phi(\mathrm{C}) \cdot \mathrm{occur(b)} \wedge \mathrm{occur(c)} \wedge \mathrm{enabling\_condition_{BC}(a,b,c)}$

Jeśli z warunków opisanych w wyrażeniach $\mathrm{enabling\_condition}$ wynika, że $\mathrm{enabling\_condition_{BC}(a,b,c)} \Longrightarrow \mathrm{enabling\_condition_B(a,b)}$ albo że $\mathrm{enabling\_condition_{BC}(a,b,c)} \Longrightarrow \mathrm{enabling\_condition_C(a,c)}$ (a wiemy, że $\exists b \in \phi(\mathrm{B}), \mathrm{c} \in \phi(\mathrm{C}) \cdot \mathrm{occur(b)} \wedge \mathrm{occur(c)} \Rightarrow \exists b \in \phi(\mathrm{B}) \cdot \mathrm{occur(b)}$ oraz $\exists b \in \phi(\mathrm{B}), \mathrm{c} \in \phi(\mathrm{C}) \cdot \mathrm{occur(b)} \wedge \mathrm{occur(c)} \Rightarrow \exists c \in \phi(\mathrm{C}) \cdot \mathrm{occur(c)}$), to definicję tę można zapisać jako

(xli) $\mathrm{M}\big(OR(A,\{B,C\})\big) \equiv \forall_{a\in\phi(A)} occur(a) \Longrightarrow \exists b \in$
$\phi(\mathrm{B}) \cdot \mathrm{occur(b)} \wedge \mathrm{enabling\_condition_B(a,b)} \vee \exists c \in$
$\phi(\mathrm{C}) \cdot \mathrm{occur(c)} \wedge \mathrm{enabling\_condition_C(a,c)}$

bez utraty informacji. W przeciwnym razie wystąpienie łączne zdarzeń trzeba zamodelować jak w punkcie 2.

Należy zwrócić uwagę, że powyższe założenie dotyczy tylko definicji minimalności zdarzeń. Przedstawiony powyżej algorytm umożliwia analizowanie bramek *OR* zdefiniowanych jak w równaniu (xl), niemniej uzyskane w ten sposób Zbiory Przyczyn będą nieminimalne według klasycznej definicji.

Przyjęcie definicji bramki (xl) za poprawną wymagałoby przyjęcia innej definicji minimalności zbiorów przyczyn, która poza zdarzeniami brałaby również pod uwagę warunki czasowe zdefiniowane dla każdego ze zbiorów przyczyn. Wymagałoby to również przeprowadzenia dodatkowych działań przy przetwarzaniu wyników analizy dla zapewnienia spełnienia nowego warunku minimalności.

Ponieważ wyspecyfikowanie wpływu łącznego wystąpienia zdarzeń wejściowych bramki *OR* na jej wyjście poprzez dodanie bramki *AND* wymaga powielenia poddrzew podpiętych do wejść rozważanej bramki, analiza wpływu iloczynu zdarzeń wejściowych bramki *OR* wyspecyfikowanego w definicji bramki będzie brana pod uwagę w dalszej części rozprawy jako podejście alternatywne. Sytuacja, w której taka definicja jest wymagana opisana została w [GW97].

Należy również zauważyć, że ponieważ wprowadzono możliwość stosowania alternatywy w definicjach bramek oraz w znormalizowanej reprezentacji wyrażeń ECSDM funkcja $enabling\_condition$ poszczególnych MZP może zawierać alternatywne warunki czasowe wyrażone w normalnej postaci sumacyjnej (DNF). Każdy iloczyn może zostać zaprezentowany jako oddzielny T-graf, choć nie każdy z nich musi być znaczący. Dany iloczyn wyrażenia DNF jest nieznaczący jeśli można go usunąć z $enabling\_condition$ danego MZP bez zmiany wartościowania całości wyrażenia dla każdego dopuszczalnego zachowania systemu (dla wszystkich możliwych funkcji $Time$). Można to wyrazić jako:

(xlii) $\forall_{Time} enabling\_condition \Leftrightarrow enabling\_condition'$,

gdzie $enabling\_condition'$ oznacza funkcję po usunięciu rozpatrywanego iloczynu.

W związku z tym wprowadzono pojęcie nadmiarowego T-grafu. T-graf nadmiarowy to T-graf, który można usunąć ze zbioru T-grafów konkretnego MZP bez zmiany wartościowania sumy tych zbiorów. W sposób oczywisty

(xliii) $\left(\exists_{T-graf2 \in MZP} \cdot (T-graf1 \Longrightarrow T-graf2)\right) \Longrightarrow$
$jest\_nadmiarowy(T-graf1, MZP)$,

gdzie $T-graf1 \in MZP$.

Dodatkowo należy zaznaczyć, że ze względu na przechodniość implikacji, nie ma potrzeby rozważania, czy graf $T-graf2$ jest nadmiarowy dla określenia nadmiarowości $T-graf1$.

Ponadto może się zdarzyć sytuacja w której dwa iloczyny będą miały wspólną część taką, że

(xliv) $T-graf3 = a \wedge b, T-graf4 = a \wedge c$ oraz $\forall_{Time}(b \vee c) = true$.

W takiej sytuacji graf $T-graf5 = a$ będzie nazywany grafem wywiedzionym z grafów $T-graf3$ oraz $T-graf4$.

Dodanie takiego grafu do zbioru T-grafów konkretnego MZP nie będzie zmieniało wartościowania $enabling\_condition$, ponieważ $\forall_{Time} T-graf5 \Leftrightarrow a \Leftrightarrow a \wedge (b \vee c) \Leftrightarrow a \wedge b \vee a \wedge c \Leftrightarrow T-graf3 \vee T-graf4$

Ponadto po dodaniu $T-graf5$ uzyskamy $jest\_nadmiarowy(T-graf3, MZP)$ oraz $jest\_nadmiarowy(T-graf4, MZP)$ co wynika odpowiednio z $a \wedge b \Longrightarrow a$ i $a \wedge c \Longrightarrow a$.

Dla T-grafów owe własności można przetłumaczyć na zależności opisane przez składowe T-grafów.

Przyjęto notację

(xlv) $e \leq e2 \Leftrightarrow start(e) = start(e2) \wedge end(e) = end(e2) \wedge weight(e) \leq weight(e2).$

Innymi słowy porównywać można tylko krawędzie o takich samych początkach i końcach. Wynikiem porównania jest wtedy wynik porównywania wag (zasadę tą przyjęto również dla pozostałych operatorów porównania).

Można udowodnić, że:

(xlvi) $(T-graf1 \Rightarrow T-graf2) \Leftrightarrow \forall_{e2 \in edges(T-graf2)} \exists_{e \in edges(T-graf1)} e \leq e2$

Wynika to z dwóch lematów:

(xlvii) $(\forall_{i=1}^{n} a_i \Longrightarrow b_i) \Longrightarrow \left(\bigwedge_{j=1}^{m} a_j \Longrightarrow \bigwedge_{i=1}^{n} b_i\right)$, gdzie $m \geq n$ oraz $a_i = a_j$ dla $i = j$

(xlviii) $\forall_{e,e2 \in edges(\cdot)} e \leq e2 \Leftrightarrow (e \Longrightarrow e2)$

Lemat (xlvii) jest łatwy do uzasadnienia. Jeśli $\bigwedge_{j=1}^{m} a_j$ jest spełnione, to spełnione są wszystkie wyrażenia a_j. Ponieważ każde z wyrażeń iloczynu $\bigwedge_{i=1}^{n} b_i$ jest implikowane przez pewne wyrażenie z iloczynu $\bigwedge_{j=1}^{m} a_j$, to wszystkie b_i również są spełnione, a więc $\bigwedge_{i=1}^{n} b_i$ jest spełnione. Stąd $\bigwedge_{j=1}^{m} a_j$ implikuje $\bigwedge_{i=1}^{n} b_i$.

Dowód lematu (xlviii) znajduje się w raporcie [Gol17a].

Jeśli krawędź e przedstawić jako trójkę uporządkowaną $\left(Time(e_s), (t, r), Time(e_e)\right)$, gdzie $e_s, e_e \in T$ i są to odpowiednio początek i koniec krawędzi, $t \in \mathbb{R}$ i jest długością krawędzi, $r \in \{>, \geq\}$ jest relacją przyporządkowaną krawędzi, a $Time: T \to \mathbb{R}$ jest możliwym zachowaniem systemu to negację krawędzi e definiuje się jako:

(xlix) $\overline{e} \stackrel{\text{def}}{=} \left(Time(e_e), (-t, \overline{r}), Time(e_s)\right),$

Gdzie operacja $\overline{\cdot}: \{>, \geq\} \to \{>, \geq\} \stackrel{\text{def}}{=} \{(>, \geq), (\geq, >)\}$

Przykład 11.

> Dla krawędzi $e = start(x) + 7 > end(y)$ jej negacja to $\overline{e} = end(y) - 7 \geq start(x) = start(x) + 7 \leq end(y)$ i jak łatwo zauważyć suma wag obu krawędzi wynosi $(0, >)$.

Można też zauważyć, że $\forall_e e \vee \overline{e} = true$. Dowód tego twierdzenia umieszczono w raporcie [Gol17a]. Łatwo również zauważyć, że prawdziwe jest wyrażenie $\forall_{a: \overline{e} \Rightarrow a} e \vee a$. Wynika to z faktu, że zawsze

gdy spełnione jest $\overline{e}$ to spełnione jest również a, a więc gdy niespełnione jest e to spełnione jest a. Czyli

(l) $\quad \overline{e} \leq a \Rightarrow \forall_{Time:T\to\mathbb{R}} e \vee a$

Dzięki (l) można zdefiniować z jakich T-grafów można wywieść nowy T-graf.

(li) $\quad da\_się\_wywieść(T-graf1, T-graf2) \Leftrightarrow vertexes(T-graf1) = vertexes(T-graf2) \wedge \left(\exists_{e\in edges(T-graf1), e2\in edges(T-graf2)} \overline{e} \leq e2 \wedge edges(T-graf1) \setminus \{e\} = edges(T-graf2) \setminus \{e2\}\right)$

Wtedy

(lii) $\quad wywiedziony(T-graf3, \{T-graf1, T-graf2\}) = (vertexes(T-graf1), edges(T-graf1) \setminus \{e\}) = (vertexes(T-graf2), edges(T-graf2) \setminus \{e2\}).$

Przyjęto przy tym, że T-graf reprezentowany jest przez uporządkowaną parę zbiorów swoich wierzchołków i krawędzi: $T-graf \stackrel{\text{def}}{=} \left(vertexes(T-graf), edges(T-graf)\right)$.

Wymaganie $vertexes(T-graf1) = vertexes(T-graf2)$ w $da\_się\_wywieść(T-graf1, T-graf2)$ wynika z faktu, że w T-grafach tworzonych na podstawie funkcji M bramek drzewa niezdatności dla każdego wierzchołka przypada co najmniej jedna krawędź (predykat $occur(x)$). Wymaganie to jest więc nadmiarowe względem wymagań odnośnie krawędzi grafów, ale zostało uwzględnione, gdyż ilustruje dlaczego można w wywiedzionym grafie przyjąć zbiór wierzchołków dowolnego z grafów wejściowych.

Z wyrażenia (li) da się łatwo wywnioskować, że dla każdej pary T-grafów można wywieść co najwyżej jeden T-graf.

6.1 Algorytm usuwania nadmiarowości REDUCE_MZP

Na podstawie rozważań przedstawionych w rozdziale 6 można zdefiniować algorytm REDUCE_MZP usuwania nadmiarowości w wyrażeniach $enabling\_condition$ poszczególnych MZP.

Algorytm ten przebiega następująco:

ALGORYTM

REDUCE_MZP

KROKI

1. Dla każdej pary T-grafów należącej do MZP wywiedź, jeśli to możliwe (patrz równanie (li)), nowy T-graf.
2. Powtarzaj dopóki udaje się wywieść co najmniej jeden nowy T-graf:
   a. Jako zbiór A przyjmij wszystkie T-grafy (wejściowe oraz wywiedzione we wszystkich poprzednich krokach).
   b. Jako zbiór B przyjmij T-grafy wywiedzione w ostatnim powtórzeniu kroku 2 (lub w kroku 1 dla pierwszego powtórzenia kroku 2).
   c. Dla każdej pary z $A \times B$ (ignorując kolejność w parze) wywiedź, jeśli to możliwe, nowy T-graf.
3. Jako zbiór A przyjmij wszystkie T-grafy (wejściowe oraz wywiedzione we wszystkich poprzednich krokach).
   Dla każdej pary $(T-graf1, T-graf2) \in A \times A$, jeśli $(T-graf1 \Longrightarrow T-graf2)$ odrzuć $T-graf1$ z puli wynikowej.

Powracając do definicji minimalności, w tym miejscu można rozważyć wpływ alternatywnej definicji minimalności na przedstawiony powyżej algorytm redukcji nadmiarowości w MZP.

Definicja minimalności umożliwiająca definiowanie w bramkach typu *OR* zależności czasowych dla wspólnego wystąpienia zdarzeń wyjściowych miałaby postać:

> ***Zbiór przyczyn z zależnościami czasowymi jest minimalny jeśli, po zastąpieniu dowolnego z należących do niego wyrażeń czasowych przez jego negację, wystąpienie opisanej przez niego sekwencji zdarzeń nie umożliwi wystąpienia zagrożenia.***

Definicja ta rozszerza definicję minimalności podaną w podrozdziale 2.3.1 o nowe warunki, nie znosząc przy tym dotychczasowych. Wystąpienie akcji dowolnego zdarzenia w wyrażeniach czasowych reprezentowane jest jako wyrażenie $occur(x)$, jest więc możliwe zastąpienie tego wyrażenia przez jego negację, co będzie odpowiadało usunięciu jednego ze zdarzeń zbioru w klasycznej definicji minimalności.

Jednak możliwe jest zastosowanie tego warunku wobec dowolnego innego wyrażenia w $enabling\_condition$ danego MZP. To powoduje, że może wystąpić Minimalny Zbiór Przyczyn, który (w sensie zawartych w nim zdarzeń, a nie zależności czasowych) jest nadzbiorem innego MZP, jednak z powodu opisanych w nim zależności czasowych jest minimalny.

Z tego powodu algorytm usuwania nadmiarowości w wynikowych wyrażeniach czasowych musiałby zostać zmodyfikowany. Najważniejszą zmianą byłoby rozpatrywanie nadmiarowości T-grafów w

oderwaniu od MZP. Algorytm ten nazwano REDUCE_MZP_ext i przebiega on w następujących krokach:

ALGORYTM

REDUCE_MZP_ext

KROKI

1. Niech A będzie zbiorem wszystkich T-grafów otrzymanych na podstawie analizy MZP danego drzewa niezdatności.
2. Niech B będzie pustym zbiorem T-grafów.
3. Dla każdej pary grafów należącej do $(T-graf1, T-graf2) \in A \times A$, jeśli $vertexes(T-graf1) = vertexes(T-graf2)$ wywiedź nowy graf (jeśli się da (patrz równanie (li))) i dodaj do zbioru B.
4. Zastąp A przez $A \cup B$.
5. Dopóki $B \neq \emptyset$ powtarzaj,
   a. Niech C będzie zbiorem pustym.
   b. Dla każdej pary grafów należącej do $(T-graf1, T-graf2) \in A \times B$, jeśli $vertexes(T-graf1) = vertexes(T-graf2)$ wywiedź nowy graf (jeśli się da) i dodaj do zbioru C.
   c. Zastąp A przez $A \cup B$.
   d. Zastąp B przez C.
6. Dla każdej pary $(T-graf1, T-graf2) \in A \times A$, jeśli $vertexes(T-graf2) \subseteq vertexes(T-graf1)$ sprawdź, czy $(T-graf1 \Longrightarrow T-graf2)$. Jeśli tak odrzuć $T-graf1$ z puli wynikowej $(A := A \setminus \{T-graf1\})$.

Porównując definicję algorytmu REDUCE_MZP oraz REDUCE_MZP_ext można zauważyć, że definicja drugiego algorytmu jest bardziej skomplikowana, gdyż nie ograniczono się do T-grafów należących do jednego MZP. Niemniej warunek $vertexes(T-graf1) = vertexes(T-graf2)$ w istocie to oznacza. A więc różnica w algorytmach sprowadza się do sprawdzania czy można odrzucić T-graf dla większej liczby par grafów - $vertexes(T-graf2) \subseteq vertexes(T-graf1)$ w REDUCE_MZP_ext zamiast $vertexes(T-graf1) = vertexes(T-graf2)$ w REDUCE_MZP.

6.2 Algorytmy MZPCALC_CONTROLABLE i MZPCALC_PAIR

Jeśli celem przeprowadzanej analizy drzewa niezdatności byłoby tylko określenie wymagań czasowych wobec systemu, można w analizie MZP zrezygnować z obliczania $enabling\_condition$ dla niektórych z nich. Dokładniej tych, dla których wymagań czasowych nie da się określić. Są to MZP nie zawierające zdarzenia kontrolowalnego. Rezygnacja z obliczania $enabling\_condition$ dla tych MZP przyśpieszy przetwarzanie drzewa. W ten sposób uzyskujemy algorytm MZPCALC_CONTROLABLE.

Dodatkowe przyspieszenie przetwarzania można uzyskać poprzez rezygnację z przetwarzania MZP zawierających tylko jedno zdarzenie kontrolowalne a poza tym tylko zdarzenia nieobserwowalne. W takich przypadkach jedynymi możliwymi do uzyskania kandydującymi wymaganiami czasowymi

wobec systemu jest konieczność występowania zdarzenia kontrolowalnego z zadaną częstotliwością. Taka modyfikacja algorytmu MZPCALC_FULL prowadzi do utworzenia algorytmu MZPCALC_PAIR.

Przebieg obu algorytmów jest bardzo podobny do algorytmu MZPCALC_FULL i składa się z następujących kroków:

ALGORYTM

MZPCALC_CONTROLABLE

KROKI

1. Dla każdego zdarzenia nie będącego zdarzeniem prostym oblicz ile zdarzeń prostych poszczególnych typów jest jego potomkami.
2. Znormalizuj wyrażenia bramki szczytowej drzewa niezdatności zgodnie z [Gol17b] (patrz Tabela 5.1), sprowadź do normalnej postaci sumacyjnej (DNF) a każdy iloczyn przedstaw w postaci T-grafu.
3. Oblicz najlżejsze ścieżki we wszystkich z utworzonych grafów po czym usuń z nich tranzycje zdarzenia szczytowego. Jeśli nie uzyskano żadnego T-grafu przerwij algorytm.
4. Do pustej listy FIFO dodaj wszystkie bramki znajdujące się na wejściu bramki szczytowej (bramki, których zdarzeniami wyjściowymi są zdarzenia wejściowe bramki szczytowej).
5. Dopóki lista nie będzie pusta powtarzaj:
    a. Pobierz pierwszą bramkę z listy,
    b. Do listy dodaj wszystkie bramki będące na wejściu pobranej bramki,
    c. Znormalizuj wyrażenia czasowe tej bramki, sprowadź do DNF i przedstaw w postaci T-grafów. Jeśli nie uzyskano żadnego T-grafu, przerwij algorytm. Oblicz najlżejsze ścieżki w uzyskanych T-grafach.
    d. Oblicz produkt kartezjański dotychczasowych T-grafów i T-grafów uzyskanych w poprzednim punkcie.
    e. Każdą parę grafów, które zawierają wspólne zdarzenie, połącz w jeden graf. Oblicz w nim najlżejsze ścieżki po czym usuń zdarzenie wspólne dla obu tworzących grafów. Odrzuć te z utworzonych T-grafów, które wśród zdarzeń w nich zawartych oraz ich potomków nie posiadają co najmniej jednego zdarzenia kontrolowalnego.
    f. Dla par grafów, które nie miały wspólnych zdarzeń pozostaw dotychczasowy T-graf (a nie graf z punktu 5.c.).
6. Z powstałej listy usuń T-grafy nadmiarowe (patrz algorytm REDUCE_MZP w rozdziale 6.1).

ALGORYTM

MZPCALC_PAIR

KROKI

1. Dla każdego zdarzenia nie będącego zdarzeniem prostym oblicz ile zdarzeń prostych poszczególnych typów jest jego potomkami.
2. Znormalizuj wyrażenia bramki szczytowej drzewa niezdatności zgodnie z [Gol17b] (patrz Tabela 5.1), sprowadź do normalnej postaci sumacyjnej (DNF) a każdy iloczyn przedstaw w postaci T-grafu.
3. Oblicz najlżejsze ścieżki we wszystkich z utworzonych grafów po czym usuń z nich tranzycje zdarzenia szczytowego. Jeśli nie uzyskano żadnego T-grafu przerwij algorytm.
4. Do pustej listy FIFO dodaj wszystkie bramki znajdujące się na wejściu bramki szczytowej (bramki, których zdarzeniami wyjściowymi są zdarzenia wejściowe bramki szczytowej).
5. Dopóki lista nie będzie pusta powtarzaj:
    a. Pobierz pierwszą bramkę z listy,
    b. Do listy dodaj wszystkie bramki będące na wejściu pobranej bramki,
    c. Znormalizuj wyrażenia czasowe tej bramki, sprowadź do DNF i przedstaw w postaci T-grafów. Oblicz najlżejsze ścieżki w uzyskanych T-grafach. Jeśli nie uzyskano żadnego T-grafu, przerwij algorytm.
    d. Oblicz produkt kartezjański dotychczasowych T-grafów i T-grafów uzyskanych w poprzednim punkcie.
    e. Każdą parę grafów, które zawierają wspólne zdarzenie, połącz w jeden graf. Oblicz w nim najlżejsze ścieżki po czym usuń zdarzenie wspólne dla obu tworzących grafów. Odrzuć te z utworzonych T-grafów, które wśród zdarzeń w nich zawartych oraz ich potomków nie posiadają co najmniej jednego zdarzenia kontrolowalnego i co najmniej jednego różnego od niego zdarzenia innego niż nieobserwowalne.
    f. Dla par grafów, które nie miały wspólnych zdarzeń pozostaw dotychczasowy T-graf (a nie graf z punktu 5.c.).
6. Z powstałej listy usuń T-grafy nadmiarowe (patrz algorytm REDUCE_MZP w rozdziale 6.1).

Ponadto można przyspieszyć obliczenia rezygnując z obliczania dla poszczególnych T-grafów zależności pomiędzy zdarzeniami nieobserwowalnymi a dowolnymi innymi, jeśli nie wpłynie to na obliczanie zależności pomiędzy zdarzeniami innymi niż nieobserwowalne.

Wykluczenie zdarzenia, którego wszystkie zdarzenia potomne są nieobserwowalne, nie wpłynie na wynikowe zależności pomiędzy zdarzeniami innymi niż nieobserwowalne, jeśli w przetwarzanym T-grafie po dodaniu dla wszystkich zdarzeń krawędzi w obu kierunkach pomiędzy ich tranzycją początkową i końcową (jeśli nie istnieją) nie istnieje ścieżka przechodząca przez obie tranzycje rozpatrywanego zdarzenia, która łączy zdarzenie kontrolowalne i inne niż nieobserwowalne (albo zdarzenia posiadające takie zdarzenia wśród swoich potomków).

Wymagane jest istnienie ścieżki przechodzącej przez obie tranzycje, ponieważ jeśli ścieżka przechodzi przez tylko jedną i jest najlżejszą ścieżką pomiędzy zdarzeniem kontrolowalnym a innym niż nieobserwowalne (albo jej fragmentem), to jej waga została już obliczona (niedopuszczalne jest występowanie pętli o wadze $(0, >)$ lub niższej, a tylko taka pętla pozwoliłaby na zmianę wyliczonej wagi).

6.3 Algorytm MZPCALC_PART

Powyższe rozważania dotyczące algorytmu MZPCALC_PAIR prowadzą do określenia algorytmu MZPCALC_PART (będącego modyfikacją MZPCALC_PAIR) jeszcze bardziej redukującego obliczenia, ale kosztem bardziej rozbudowanych testów. Składa się on z następujących kroków (przez potencjalnie przechodzące ścieżki rozumie się oczywiście ścieżki przechodzące przez zadane wierzchołki po dodaniu do T-grafu cyklów dwukrawędziowych łączących tranzycje tych samych zdarzeń):

ALGORYTM

MZPCALC_PART

KROKI

1. Dla każdego zdarzenia nie będącego zdarzeniem prostym oblicz ile zdarzeń prostych poszczególnych typów jest jego potomkami.
2. Znormalizuj wyrażenia bramki szczytowej drzewa niezdatności zgodnie z [Gol17b] (patrz Tabela 5.1), sprowadź do normalnej postaci sumacyjnej (DNF) a każdy iloczyn przedstaw w postaci T-grafu.
3. Oblicz najlżejsze ścieżki we wszystkich z utworzonych grafów po czym usuń z nich tranzycje zdarzenia szczytowego. Jeśli nie uzyskano żadnego T-grafu przerwij algorytm.
4. Do pustej listy FIFO dodaj wszystkie bramki znajdujące się na wejściu bramki szczytowej (bramki, których zdarzeniami wyjściowymi są zdarzenia wejściowe bramki szczytowej).
5. Dopóki lista nie będzie pusta powtarzaj:
    a. Pobierz pierwszą bramkę z listy,
    b. Do listy dodaj wszystkie bramki będące na wejściu pobranej bramki,
    c. Znormalizuj wyrażenia czasowe tej bramki, sprowadź do DNF i przedstaw w postaci T-grafów. Oblicz najlżejsze ścieżki w uzyskanych T-grafach. Jeśli nie uzyskano żadnego T-grafu, przerwij algorytm.
    d. Oblicz produkt kartezjański dotychczasowych T-grafów i T-grafów uzyskanych w poprzednim punkcie.
    e. Każdą parę grafów, które zawierają wspólne zdarzenie:
        i. połącz w jeden graf. Oblicz w nim najlżejsze ścieżki po czym usuń zdarzenie wspólne dla obu tworzących grafów.
        ii. Jeśli w utworzonym grafie wśród zdarzeń w nich zawartych oraz ich potomków nie ma co najmniej jednego zdarzenia kontrolowalnego i co najmniej jednego różnego od niego zdarzenia innego niż nieobserwowalne, to taki graf odrzuć.
        iii. W utworzonym T-grafie usuń te zdarzenia, które mają samych nieobserwowalnych potomków oraz przez których obie tranzycje nie przechodzą potencjalnie ścieżki łączące zdarzenie kontrolowalne oraz inne niż nieobserwowalne (albo przodków takich zdarzeń).
    f. Dla par grafów, które nie miały wspólnych zdarzeń pozostaw dotychczasowy T-graf (a nie graf z punktu 5.c.).
6. Z powstałej listy usuń T-grafy nadmiarowe (patrz algorytm REDUCE_MZP w rozdziale 6.1).

7 Wyznaczanie wymagań bezpieczeństwa w metodzie TERM

Ogólna idea wyznaczania wymagań bezpieczeństwa jest prosta: dla każdego MZP należy zidentyfikować zależności czasowe pomiędzy zdarzeniem kontrolowalnym a zdarzeniem innym niż nieobserwowalne (obserwowalne, przewidywalne, kontrolowalne), których spełnienie uniemożliwi spełnienie $enabling\_condition$ dla danego MZP. Zakładając, że $enabling\_condition$ dla danego MZP nie zawiera alternatyw, spełnienie dowolnej ze zidentyfikowanych zależności czasowych spowoduje zablokowanie możliwości wystąpienia hazardu. W razie występowania w $enabling\_condition$ alternatyw, każda alternatywa musi być „unieszkodliwiana" osobno.

W prezentowanym podejściu identyfikacja zależności czasowych odbywa się automatycznie (na podstawie klasyfikacji zdarzeń występujących w MZP), natomiast wybór zależności czasowej, która będzie „unieszkodliwiać" konkretne MZP, pozostawiony jest analitykowi. Wybrane zależności czasowe stają się wtedy czasowymi wymaganiami bezpieczeństwa wobec systemu.

Nie istnieje oczywiście gwarancja, że zaproponowane w ten sposób zależności czasowe będą możliwe do zaimplementowania w systemie (np. ze względu na konflikt takiego wymagania z misją systemu). Nie istnieje również gwarancja, że dla każdego MZP zostanie zaproponowana jakakolwiek zależność. Zapewnienie bezpieczeństwa systemu względem takich MZP będzie musiało odbywać się w sposób „tradycyjny", tzn. przez zapewnienie odpowiednio niskich prawdopodobieństw wystąpienia poszczególnych zdarzeń.

Przykład 12.

Przyjmijmy, że MZP jest określony wzorem:

(liii) $\{e_1, e_2, e_3 | e_1 \in \phi(E_1) \wedge e_2 \in \phi(E_2) \wedge e_3 \in \phi(E_3) \wedge occur(e_1) \wedge occur(e_2) \wedge occur(e_3) \wedge start(e_3) > start(e_1) \wedge start(e_1) + 7 > start(e_3) \wedge duration(e_2) > 4\}$,

oraz $E_1 \in O, E_2 \in N, E_3 \in C$.

Jeśli nie jest możliwe zapobieżenie występowaniu któregokolwiek ze zdarzeń, to aby zagrożenie nie wystąpiło należy zapewnić, że $start(e_3) \leq start(e_1)$ lub $start(e_1) + 7 \leq start(e_3)$ lub $duration(e_2) \leq 4$.

Zapewnienie, że czas trwania akcji e_2 będzie mniejszy niż 4 jednostki czasu, nie będzie możliwe, gdyż E_2 nie jest zdarzeniem kontrolowalnym. Zapewnienie, że akcja e_3 rozpocznie się przed początkiem akcji e_1 też nie będzie możliwe: aby rozpocząć akcję e_3 przed akcją e_1 trzeba o niej wiedzieć z wyprzedzeniem, a e_1 jest akcją obserwowalną. Zatem jedyną zależnością czasową, której spełnienie uniemożliwi wystąpienie zagrożenia powodowanego przez dany MZP jest $start(e_1) + 7 \leq start(e_3)$.

Zadaniem analityka jest ocena, czy możliwe jest takie zaprojektowanie systemu, aby po każdym rozpoczęciu zdarzenia E_1 zdarzenie E_3 nie rozpoczęło sie przez 7 jednostek czasu. W przypadku, gdyby dla rozpatrywanego MZP zidentyfikowano więcej niż jedną zależność czasową mogącą uniemożliwić wystąpienie tego MZP,

analityk mógłby wybrać, którą z tych zależności zaimplementować w projektowanym systemie.

Aby ograniczyć proponowanie jako ograniczeń wobec systemu zależności czasowych, które nie mogą być spełnione, wprowadzono pojęcie ograniczeń dziedzinowych. Ograniczenia dziedzinowe są specyfikowane jako *inwarianty* czyli takie wyrażenia czasowe, które są spełnione dla wszystkich zachowań systemu, a nie tylko tych dopuszczających wystąpienie zagrożenia. Są one specyfikowane wyłącznie dla zdarzeń prostych.

Ograniczenia dziedzinowe można podzielić na dwie grupy. Takie, które dotyczą tylko jednego zdarzenia prostego, oraz takie, które dotyczą większej liczby. Różnica jest istotna ze względu na fakt innego sposobu ich reprezentowania oraz wykorzystywania.

Ograniczenia dziedzinowe dotyczące pojedynczego zdarzenia ze swej natury umożliwiają jedynie definicję długości trwania akcji tego zdarzenia. Choć w narzędziu analitycznym definiowane są w bardzo podobny sposób do bramek drzewa, wyróżnia je jeden fakt. A mianowicie ograniczenia dziedzinowe nie są traktowane jako zależności czasowe warunkujące wystąpienie hazardu. Zamiast tego używane są do sprawdzania kandydujących wymagań wobec systemu oraz wykorzystywane są jako dodatkowe informacje przy przetwarzaniu MZP (z ostatecznych MZP są jednak usuwane). Ponieważ ograniczenia dziedzinowe dotyczące pojedynczego zdarzenia są wykorzystywane przy budowaniu MZP, nie jest możliwe wyznaczenie MZP będącego z nimi sprzecznym.

Ograniczenia dziedzinowe dotyczące więcej niż jednego zdarzenia prostego definiowane są osobno, w formie $warunek \Longrightarrow ograniczenie$. Zarówno $warunek$, jak i $ograniczenie$ są iloczynami wyrażeń czasowych w notacji ECSDM. Można zdefiniować dowolną liczbę ogólnych ograniczeń dziedzinowych.

Przykład 13.

Ograniczenia dziedzinowe dotyczące więcej niż jednego zdarzenia byłyby przydatne w przypadku konieczności użycia wybranego zdarzenia prostego w dwóch miejscach drzewa (na wejściu dwóch różnych bramek). Ponieważ obecnie metoda nie wspiera takiej sytuacji (patrz podrozdział 10.4), zdarzenie te musiałoby zostać zamodelowane jako dwa różne zdarzenia (przyjmijmy, że ich identyfikatory to E_1 oraz E_2). Z faktu, że te dwa identyfikatory opisują to samo zdarzenie, wynika, że ich akcje nie mogą mieć części wspólnej chyba, że są sobie równe. Odzwierciedlić tą zależność można przez następujące ograniczenie dziedzinowe:

(liv) $\forall_{e_1 \in \phi(E_1), e_2 \in \phi(E_2)} occur(e_1) \wedge occur(e_2) \wedge start(e_1) = start(e_2) \Rightarrow end(e_1) = end(e_2)$,

(lv) $\forall_{e_1 \in \phi(E_1), e_2 \in \phi(E_2)} occur(e_1) \wedge occur(e_2) \wedge start(e_1) < start(e_2) \Rightarrow end(e_1) < start(e_2)$,

(lvi) $\forall_{e_1 \in \phi(E_1), e_2 \in \phi(E_2)} occur(e_1) \wedge occur(e_2) \wedge start(e_1) > start(e_2) \Rightarrow start(e_1) > end(e_2)$.

(liv) oznacza, że jeśli akcje zdarzeń E_1 oraz E_2 rozpoczynają się w tym samym momencie, to skończyć się również muszą równocześnie. (lv) oraz (lvi) oznaczają,

że akcje zdarzeń E_1 oraz E_2 są kolejnymi akcjami jednego zdarzenia, co jest wyrażone przez wymaganie, aby ta która się zacznie pierwsza, skończyła się zanim rozpocznie się następna.

Podczas analizy drzewa niezdatności można te ograniczenia wykorzystać aby odrzucić, jako niemogących zaistnieć, MZP które wymagają, aby $\exists_{e_1 \in \phi(E_1), e_2 \in \phi(E_2)} start(e_1) = start(e_2) \wedge end(e_1) > end(e_2)$ - patrz (liv).

Ograniczenia dziedzinowe dotyczące więcej niż jednego zdarzenia umożliwiają również odrzucenie niespełnianych kandydujących wymagań bezpieczeństwa. W przypadku ograniczenia dziedzinowego

(lvii) $$\forall_{e_1 \in \phi(E_1), e_2 \in \phi(E_2)} true \Rightarrow end(e_1) = end(e_2),$$

kandydujące wymaganie bezpieczeństwa $end(e_1) + 3 < end(e_2)$ nie jest spełniane, gdyż dla wszystkich zachowań systemu akcje e_1 oraz e_2 kończą się jednocześnie.

W ogólności wyznaczanie kandydujących wymagań bezpieczeństwa opiera się na prostym wyrażeniu $\forall_e \overline{e \wedge \overline{e}}$. Dla każdej zawierającej zdarzenie kontrolowalne krawędzi T-grafu reprezentującego wybrany MZP obliczana jest krawędź zanegowana, po czym sprawdzane są dodatkowe warunki umożliwiające zaklasyfikowanie wynikowej krawędzi jako kandydującego wymagania wobec systemu. Warunki te, to możliwość spełnienia wyrażenia oraz brak sprzeczności z ograniczeniami dziedzinowymi.

Spełnialność wyrażenia czasowego rozważana jest w kontekście wprowadzonej w tej rozprawie klasyfikacji zdarzeń. Nad zdarzeniami kontrolowalnymi projektanci systemu mają kontrolę a przewidywalne są w stanie przewidzieć z wystarczającym wyprzedzeniem i zareagować. W związku z tym wszystkie zależności czasowe dotyczące dwóch zdarzeń kontrolowalnych bądź zdarzenia kontrolowalnego oraz przewidywalnego traktowane są jako potencjalnie spełniane i akceptowane jako kandydujące wymagania czasowe.

W przypadku zdarzenia obserwowalnego reakcja jest z definicji możliwa dopiero wystąpieniu zdarzenia. W związku z tym za kandydujące wymagania czasowe uznawane są tylko takie wyrażenia czasowe, które mogą przyjąć wartość prawda jeśli opisywane w nim zdarzenie kontrolowalne wystąpi po zdarzeniu obserwowalnym.

Wyrażenia zawierające zdarzenia nieobserwowalne są traktowane jako niespełnialne (wyjątek opisany w Przykład 14).

7.1 Algorytm REQCALC doboru dodatkowych wymagań czasowych

Pełen algorytm doboru proponowanych wymagań wobec systemu wygląda następująco:

ALGORYTM

REQCALC

KROKI

1. Dla każdego T-grafu reprezentującego $enabling - condition$ jednego z MZP:
    a. Określ ogólne *ograniczenia dziedzinowe*, które go dotyczą. Jako A przyjmij iloczyn wszystkich aplikowalnych $ograniczeń$.
    b. Jeśli dany T-graf jest sprzeczny z jego ograniczeniami usuń ten T-graf ze zbioru wynikowych MZP i przejdź do następnego T-grafu.
    c. Do iloczynu A dołącz ograniczenia dziedzinowe specyficzne dla wszystkich zdarzeń występujących w danym MZP.
    d. Dla każdej krawędzi e występującej w rozważanym T-grafie oblicz $\overline{e}$.
    e. Dla każdej krawędzi $\overline{e}$:
        i. sprawdź, czy:
            1. Zawiera zdarzenie kontrolowalne.
            2. Nie jest sprzeczna z A.
        ii. Krawędź niespełniającą powyższych warunków odrzuć i przejdź do następnej.
        iii. sprawdź czy:
            1. Drugie zdarzenie jest również zdarzeniem kontrolowalnym bądź przewidywalnym,
            2. Drugie zdarzenie jest zdarzeniem obserwowalnym oraz wyrażenie $\overline{e}$ nie wymaga, aby wystąpiło ono po zdarzeniu kontrolowalnym.
        iv. Krawędź spełniającą powyższe warunki dodaj do zbioru B proponowanych wymagań systemowych powiązanych z danym T-grafem.
        v. Sprawdź czy rozważana krawędź dotyczy zdarzenia nieobserwowalnego oraz wymaga aby odstęp pomiędzy tym zdarzeniem, a zdarzeniem kontrolowalnym nie był większy niż zadana wartość.
        vi. W tym przypadku czas wystąpienia tranzycji zdarzenia nieobserwowalnego zastąp przez czas wystąpienia tranzycji zdarzenia kontrolowalnego występującego w tym wyrażeniu. Podstawiona tranzycja powinna być tego samego typu jak w drugim wierzchołku, jednak dotyczyć innej akcji tego samego zdarzenia. Tak stworzoną krawędź dodaj do zbioru B.
    f. Dowolne wyrażenie zawarte w zbiorze B, jeśli zapewnić jego spełnienie, uniemożliwia wystąpienie zagrożenia w scenariuszu identyfikowanym przez dany T-graf.

Przy czym:

Ad. 1.a. Ogólne ograniczenia dziedzinowe dotyczące danego T-grafu to te z wyspecyfikowanych ograniczeń dotyczących wielu zdarzeń, które nie mają $warunku$ obowiązywania ograniczenia, bądź których $warunek$ wystąpienia jest implikowany przez T-graf.

Ad. 1.b. T-graf jest sprzeczny z ograniczeniami, jeśli po przedstawieniu ograniczeń w postaci krawędzi i dodaniu ich do T-grafu powstaje graf sprzeczny (zawierający cykl ujemny lub o wadze $(0, >)$).

Ad. 1.e.i.2. Krawędź jest sprzeczna ze zbiorem ograniczeń dziedzinowych A, jeśli przy wykonywaniu algorytmu PTGRAF na T-grafie zbudowanym z tych ograniczeń oraz kandydującej krawędzi $\overline{e}$ zostanie wykryty cykl o wadze $(0, >)$ lub lżejszej.

Przykład 14.

Ad. 1.e.vi. W jednym z MZP dla systemu palnika gazowego opisanego w podrozdziale 9.1.2.1 występuje zależność $start(e_{14}) + 10 < end(e_7)$. W drzewie niezdatności tego systemu zdarzenie E_7 oznacza otwarcie zaworu gazu w wyniku awarii kontrolera, a E_{14} brak sygnału zapłonu do zapalniczki. Wyrażenie to oznacza więc, że hazard nastąpi jeśli przez więcej niż 10 jednostek czasu zawór gazu będzie otwarty w wyniku awarii kontrolera a w tym czasie nie będzie iskry z zapalniczki. Negacja tej zależności to $start(e_{14}) + 10 \geq end(e_7)$. Wyrażenie to spełnia warunek 1.e.v., a po przekształceniu zgodnie z punktem 1.e.vi. otrzymujemy wyrażenie:

(lviii) $$\forall_{e_{14} \in E_{14}} \exists_{e`_{14} \in E_{14}} \cdot start(e_{14}) + 10 \geq end(e`_{14}),$$

Które można przetłumaczyć jako wymaganie, aby w rozpatrywanym systemie co najmniej raz na 10 jednostek czasu uruchamiać zapalniczkę.

Łączna lista zidentyfikowanych wyrażeń czasowych stanowi punkt wyjścia do pracy analityka, który z zaproponowanych wyrażeń czasowych wybiera te, które mogą zostać w systemie zaimplementowane. Metoda ta jest więc metodą wstępnego odfiltrowania wyrażeń czasowych, na spełnialność których projektanci systemu nie mają wpływu.

8 Metoda walidacji uzyskanych wyników

8.1 Cele i sposób walidacji

Przeprowadzone działania walidacyjne miały na celu ilościową ocenę zaproponowanych algorytmów ograniczania zakresu analizy Minimalnych Zbiorów Przyczyn oraz wywodzenia kandydujących czasowych wymagań bezpieczeństwa wobec analizowanego systemu.

Walidacja została przeprowadzona na podstawie drzew niezdatności opracowanych dla czterech systemów:

- System palnika gazowego
- Przejazd kolejowy
- System monitoringu ruchu
- Rozjazd kolejowy

Dla każdego analizowanego systemu mierzone metryki zostały określone na podstawie metody GQM (ang. Goal/Question/Metrics) [SB99]. Istotą tej metody jest sterowanie zakresem pomiarów przy użyciu podejścia zstępującego (ang. top-down). Definiowanie pomiaru prowadzone jest na trzech poziomach:

1. Cel - ma jasno określać cel pomiaru, jego przedmiot oraz kontekst. Cel określany jest na wysokim poziomie abstrakcji.
2. Pytanie - pytania są przypisywane do wybranego celu i mają wyjaśnić w jaki sposób zrealizowany będzie cel pomiarów.
3. Metryka - metryka stanowi częściową odpowiedź na pytanie i ma charakter ilościowy. Ma charakter szczegółowy, mówi nie tylko co należy zmierzyć, ale w wypadku gdy mierzona wielkość się zmienia, określa również moment pomiaru.

Przeprowadzone pomiary miały na celu wykazanie tezy:

> ***Proponowana metoda może być efektywnie zastosowana w odniesieniu do rzeczywistych przypadków komputerowych systemów związanych z bezpieczeństwem.***

Zaplanowano przeprowadzenie dwu studiów przypadku. W obu w celu określenia efektywności opracowanych algorytmów niezbędnym było porównanie czasów przetwarzania poszczególnych drzew niezdatności z ich rozmiarem. W celu określenia potencjału dalszej redukcji czasów przetwarzania dla każdego przypadku mierzony był również rozmiar uzyskanego rozwiązania.

Przeprowadzenie tak zaprojektowanych badań wymagało przeprowadzenia dwóch typów pomiarów:

- pomiary określające rozmiar problemu i rozmiar rozwiązania,

- pomiary określające czasowy koszt uzyskania rozwiązania.

Pomiary mające na celu określenie wielkości problemu oraz rozwiązania (po eliminacji nadmiarowych MZP) były wykonywane na podstawie analizy dokumentów (właściwego drzewa niezdatności wraz z Minimalnymi Zbiorami Przyczyn oraz kandydującymi wymaganiami czasowymi). Dokumenty te zapisane były w formacie XML [XML08], a do ich przeglądania użyto narzędzia PolymorphFT2 (zobacz podrozdział 8.2).

Pomiary mające na celu określenie kosztu uzyskania rozwiązania uzyskano przy użyciu tego samego narzędzia, poprzez pomiar czasu wykonywania algorytmów analizy drzewa niezdatności oraz wywodzenia proponowanych wymagań czasowych. Czas ten określany był na podstawie czasu systemowego wypisywanego przez narzędzie do pliku tekstowego przed rozpoczęciem i po zakończeniu przetwarzania. W pliku tym zapisywany był również rozmiar rozwiązania - liczba MZP oraz łączna liczba wyrażeń we wszystkich MZP przed redukcją nadmiarowych MZP. Czas wyliczania powyższych wartości nie był wliczany do czasu działania algorytmu.

Pomiary prowadzone były na komputerze z procesorem Intel Core i7 920 o częstotliwości taktowania wynoszącej 2,67 GHz, z 6GB pamięci operacyjnej w konfiguracji (3x2GB tripple channel) oraz systemem Windows 7 Pro w wersji 64 bitowej z zainstalowanym pakietem Service Pack 1. Dla ograniczenia wpływu innych aplikacji na wynik pomiaru wyłączono aplikacje nie mające związku z pomiarami (również te działające w zasobniku systemowym (ang. systray) poza programem antywirusowym). Pomiary czasów analizy były wykonywane wielokrotnie, liczba powtórzeń określana była dla każdego studium przypadku osobno.

W każdym przypadków pomiary wykonywano zarówno z wykorzystaniem algorytmu redukcji nadmiarowych MZP (REDUCE_MZP) oraz z wykorzystaniem rozszerzonego algorytmu redukcji nadmiarowych MZP (REDUCE_MZP_ext). We wszystkich studiach przypadków wersja podstawowa tego algorytmu jest wystarczająca – algorytm rozszerzony porówna dokładnie te same pary MZP dla usunięcia nadmiarowości. Różnice czasowe dla obu wersji mogą wynikać z różnego sposobu doboru par MZP do porównania.

Procedura pomiarowa za każdym razem wyglądała następująco:

1. Wczytywano dokument zawierający drzewo niezdatności będące przedmiotem eksperymentu.
2. Dwukrotnie analizowano dokument badanym typem algorytmu analitycznego.
3. Przeprowadzano zadaną w danym studium przypadku liczbę analiz przy użyciu badanego algorytmu.
4. Po zakończeniu pomiarów dane czasowe były pobierane z pliku[12].

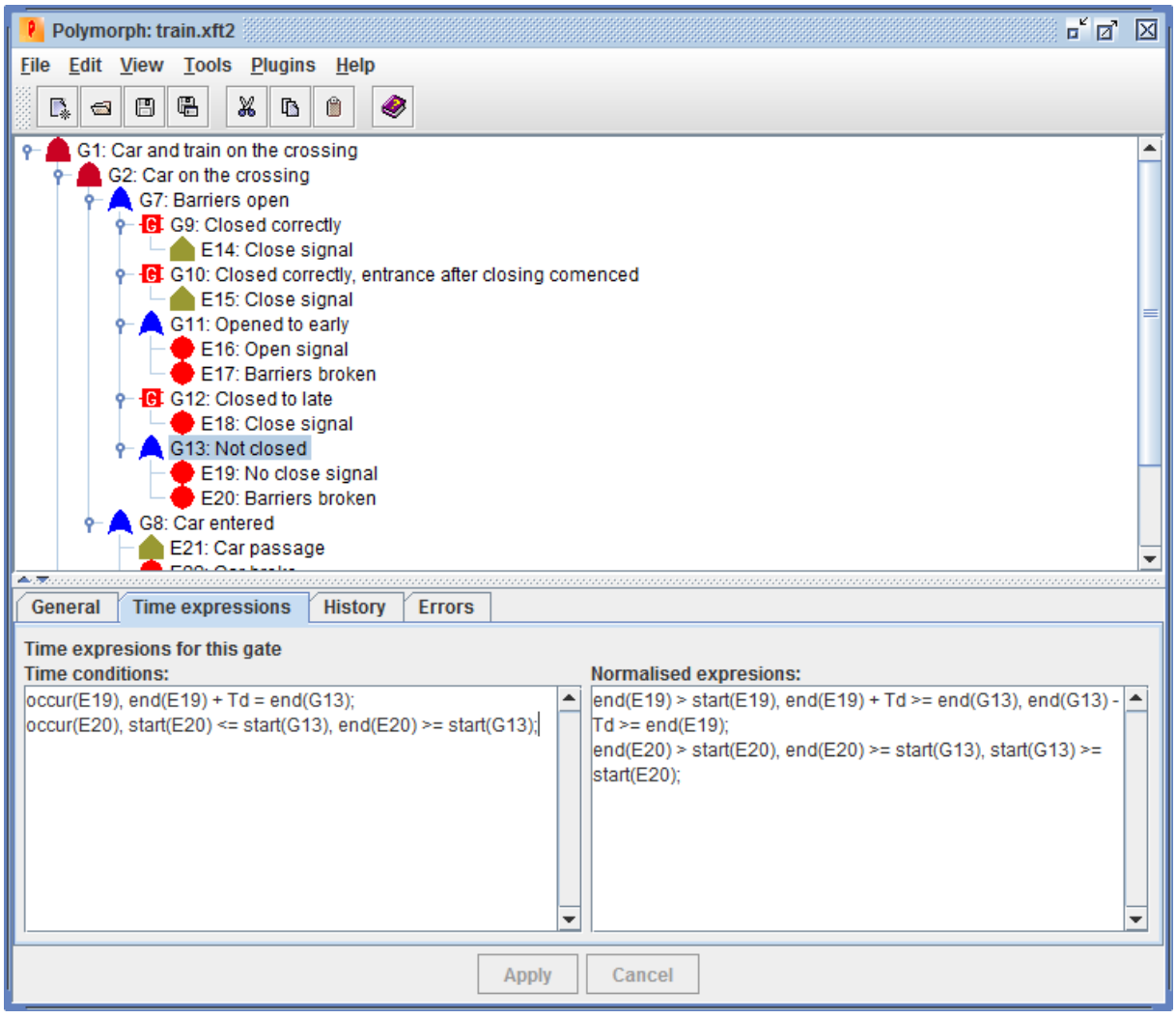
Pierwsze dwa pomiary (punkt 2) przeprowadzane były w związku z zaobserwowanym zjawiskiem zawyżania czasu wykonywania się algorytmów w pierwszym a niekiedy i drugim przebiegu w stosunku do reszty pomiarów. Uzyskane w tych pomiarach czasy wykonania algorytmu nie były brane

[12] W przypadku drugiego studium przypadku były to również dane dotyczące rozmiaru rozwiązania.

pod uwagę przy określaniu średniego czasu wykonania, pomiary te nie były również wliczane do łącznej liczby wykonanych pomiarów.

8.2 PolymorphFT2 - narzędzie wspierające stosowanie metody TREM

Metodzie TREM towarzyszy wsparcie narzędziowe - program PolymorphFT2 (jest to druga wersja narzędzia, pierwszą opisano w [Gol05]). Program ten, stworzony w celu wsparcia opisanych w niniejszym raporcie badań, umożliwia tworzenie drzew niezdatności i określanie warunków czasowych dla poszczególnych bramek drzewa. Możliwe jest również specyfikowanie ograniczeń dziedzinowych. Okno edycyjne programu przedstawiono na Rysunek 8.1.



Rysunek 8.1 Edycja drzewa niezdatności w programie PolymorphFT2

Program umożliwia również analizę stworzonych drzew niezdatności przy użyciu wszystkich opisanych w tej pracy algorytmów. Po wyznaczeniu Minimalnych Zbiorów Przyczyn można je w programie przeglądać. Dostępne są trzy widoki:

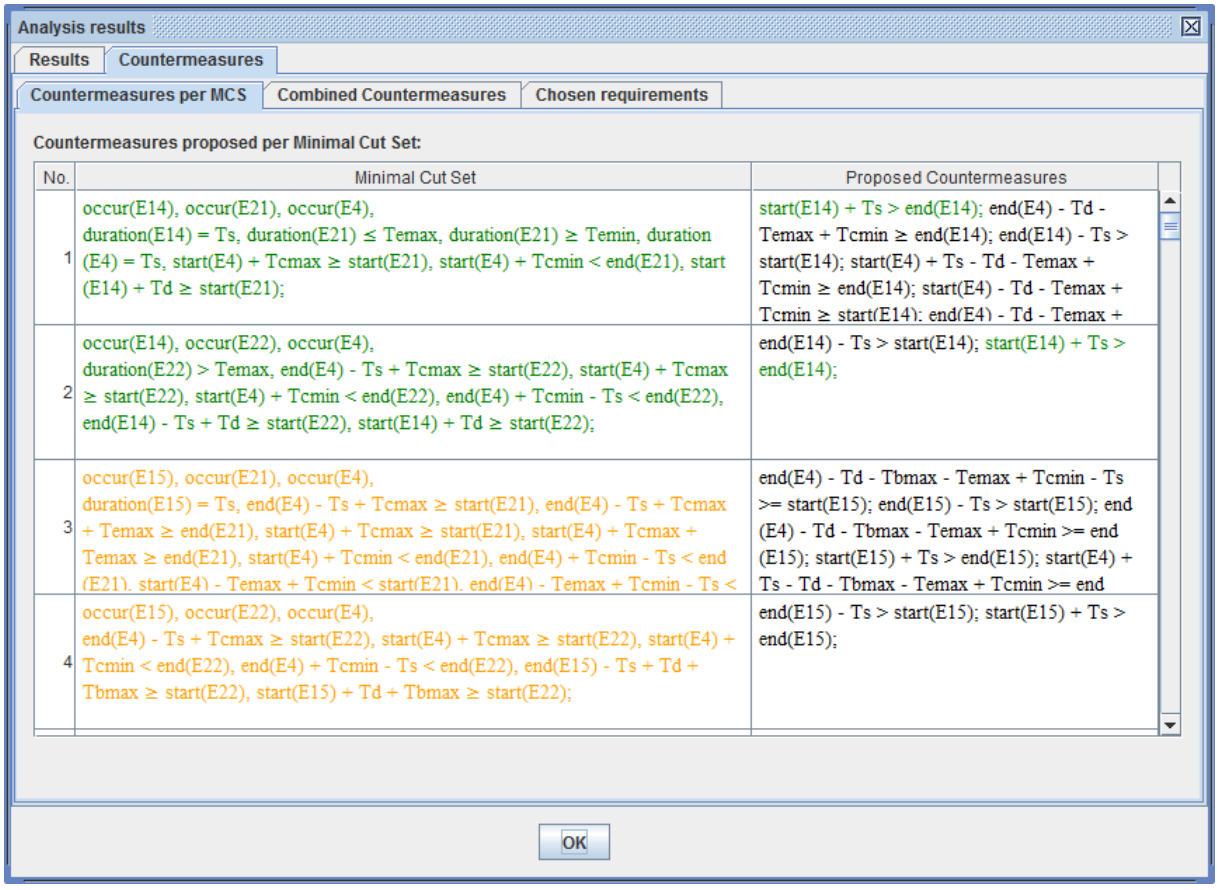
- wszystkie wyznaczone wyrażenia czasowe,
- wyrażenia czasowe ze zredukowanymi wyrażeniami nadmiarowymi,

- zinterpretowane wyrażenia czasowe.

Interpretacja jest procesem odwrotnym do normalizacji wyrażeń ECSDM, a więc polega na zastąpieniu funkcjami wyrażeń prostych.

Ponadto program udostępnia informację o proponowanych wymaganiach czasowych. W tym przypadku też dostępne są trzy widoki:

- Minimalne Zbiory Przyczyn i przypisane im proponowane wymagania czasowe. MZP bez zidentyfikowanych proponowanych wymagań czasowych wypisane są czerwoną czcionką, te ze zidentyfikowanymi wymaganiami czcionką żółtą, a te, dla których chociaż jedno z wymagań wybrano do realizacji, czcionką zieloną (patrz Rysunek 8.2),
- Zebrane proponowane wymagania czasowe i numery MZP których wystąpieniu mogą zapobiec,
- Lista wymagań czasowych przyjęta do realizacji.



Rysunek 8.2 MZP i proponowane wymagania czasowe

9 Badania walidacyjne

W celu oceny zaproponowanej metody TERM przeprowadzono szereg eksperymentów z wykorzystaniem programu PolymorphFT2. Eksperymenty te zostały ujęte w dwa studia przypadków, w ramach których rozpatrzono cztery przykładowe systemy związane z bezpieczeństwem.

9.1 CS-1 Pierwsze studium przypadku

9.1.1 Cel studium przypadku

Pierwsze studium przypadku ma na celu zbadanie, czy zaproponowana metoda umożliwia redukcję nakładów pracy analityka przy wywodzeniu wymagań czasowych wobec podsystemów sterujących analizowanego systemu, oraz ocenę zysku z ograniczenia zakresu analizy Minimalnych Zbiorów Przyczyn.

Cel i zakres studium opisano za pomocą szablonu GQM:

Zbadać metodę TREM w celu oceny redukcji nakładów pracy przez analityka przy proponowaniu wymagań czasowych wobec analizowanego systemu, uzyskiwanego wzrostu bezpieczeństwa oraz zysku z wprowadzenia algorytmów MZPCALC_CONTROLABLE, MZPCALC_PAIR oraz MZPCALC_PART względem algorytmu MZPCALC_FULL.

Pytania i metryki:

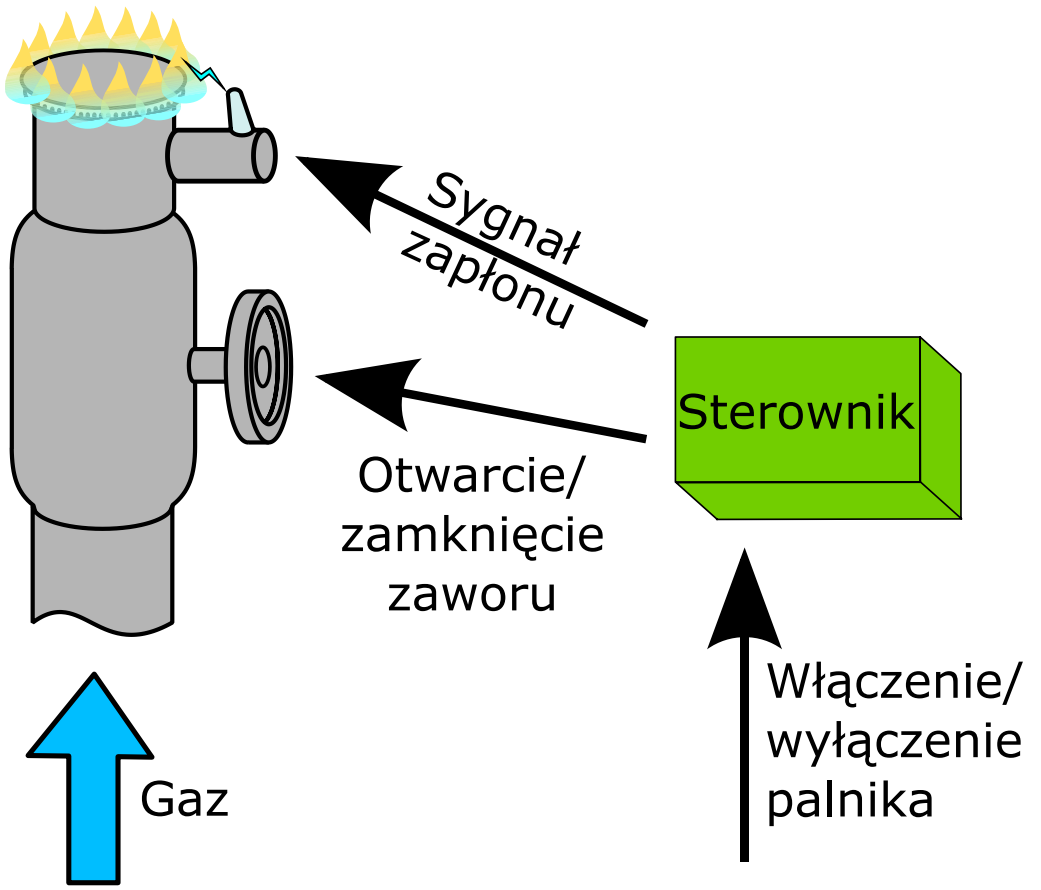
- P1: Jaki jest rozmiar problemu (rozmiar analizowanych drzew niezdatności)?
    - M1.1: Liczba bramek w drzewie niezdatności.
    - M1.2: Liczba zdarzeń prostych, z podziałem na typy.
- P2: Jaki jest rozmiar rozwiązania (produkt algorytmu wyznaczania MZP)?
    - M2.1: Liczba Minimalnych Zbiorów Przyczyn (dla poszczególnych wariantów algorytmu).
    - M2.2: Liczba wyrażeń w Minimalnych Zbiorach Przyczyn (dla poszczególnych wariantów algorytmu).
- P3: Jaki jest zysk z ograniczenia zakresu analizowanych Minimalnych Zbiorów Przyczyn?
    - M3.1: Czas analizy problemu (dla poszczególnych wariantów algorytmu).
- P4: Jaki jest zakres danych, które musi ocenić analityk, w tradycyjnej analizie TFTA oraz TREM?
    - M4.1: Liczba wyrażeń czasowych zawartych w MZP po redukcji (tradycyjna TFTA).
    - M4.2: Liczba wyrażeń czasowych zawartych w kandydujących czasowych wymaganiach bezpieczeństwa wobec systemu (TREM).
- P5: Jaki jest stopień osiągniętej poprawy bezpieczeństwa?

- M5.1: Liczba MZP po redukcji.
- M5.2: Liczba MZP, dla których zidentyfikowano co najmniej jedno kandydujące wymaganie czasowe.

9.1.2 Przedmiot badań

Badania przeprowadzono na podstawie drzew niezdatności opracowanych dla trzech systemów opisanych poniżej.

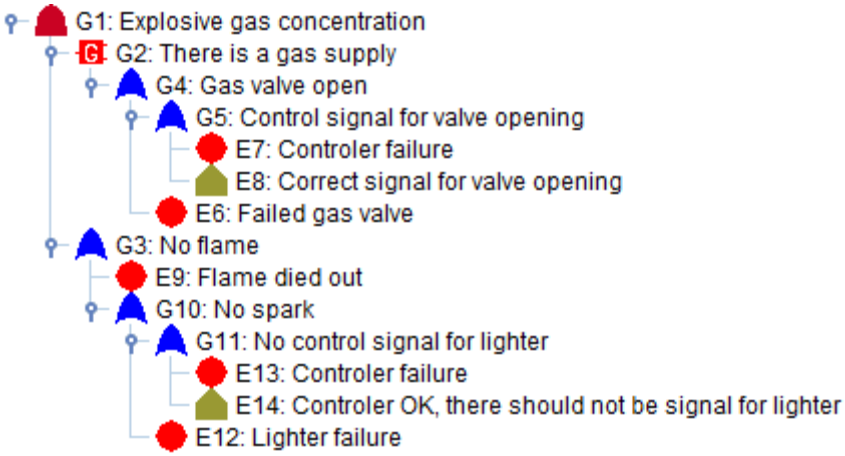
9.1.2.1 System palnika gazowego (PG)



Rysunek 9.1 Diagram systemu palnika gazowego

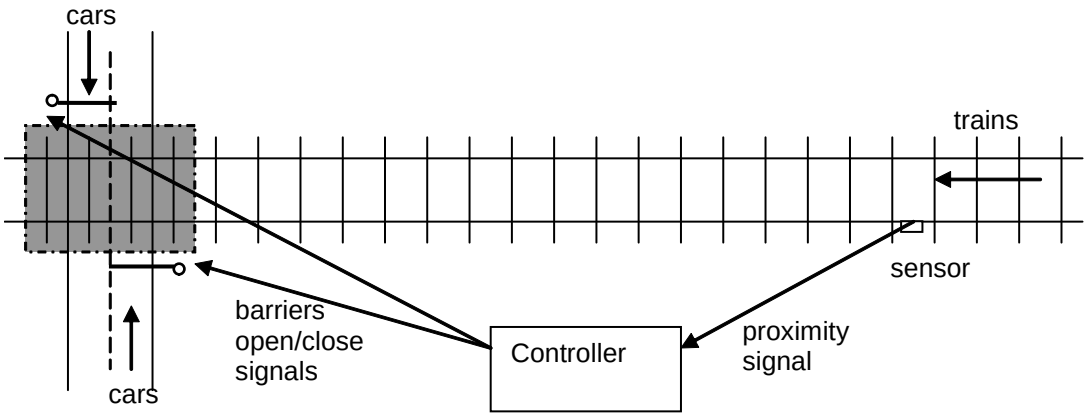
Schemat systemu przedstawia Rysunek 9.1. Można go w skrócie opisać, jako palnik gazowy z elektronicznie sterowanym zaworem gazu oraz zapalnikiem. Sterownik systemu odbiera sygnały zapalenia i wyłączenia palnika. W celu zapalenia palnika otwiera zawór gazu, a po odpowiednim czasie wysyła sygnał do zapalnika w celu wytworzenia iskry zapalającej gaz (czas od otwarcia zaworu do wysłania sygnału musi pozwolić, aby gaz osiągnął stężenie palne, ale nie wybuchowe). Osiągnięcie przez gaz stężenia wybuchowego uznawane jest za zagrożenie. Szczegółowy opis systemu palnika gazowego znajduje się m.in. w [War96].

Dla tak wyspecyfikowanego systemu opracowano drzewo niezdatności przedstawione na Rysunek 9.2:



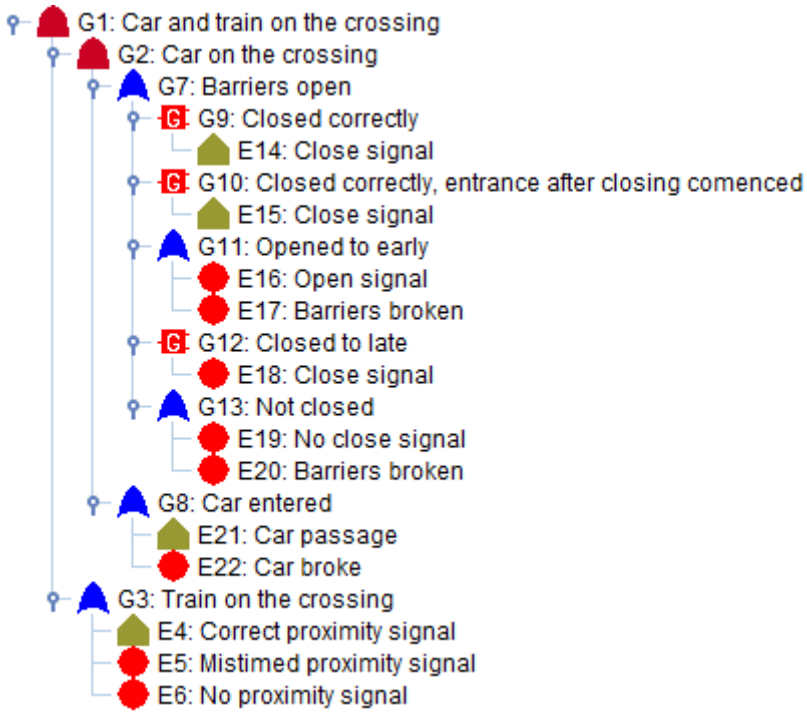
Rysunek 9.2 Drzewo niezdatności systemu palnika gazowego

9.1.2.2 Przejazd kolejowy (PK)



Rysunek 9.3 Diagram przejazdu kolejowego

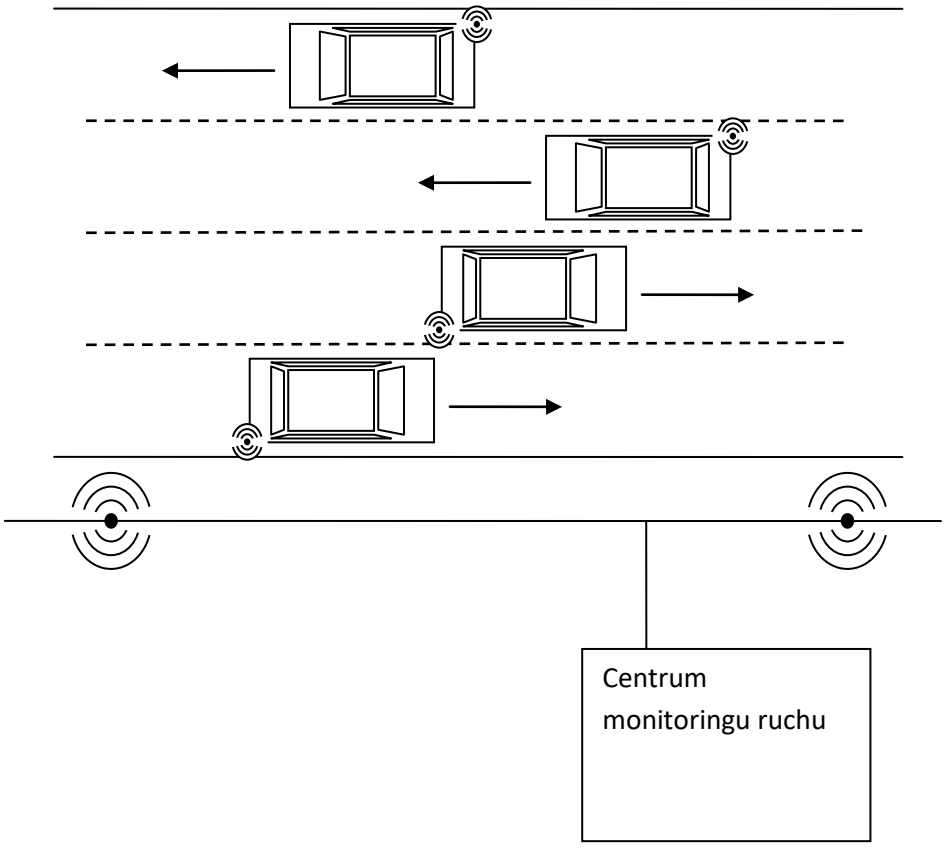
Schemat analizowanego systemu przedstawiono na Rysunek 9.3. Jest to przejazd drogi przez tor kolejowy. Tor umożliwia ruch pociągów tylko w jednym kierunku. Droga jest jednojezdniowa, dwupasmowa, z przeciwnymi kierunkami ruchu na pasach. Wjazdu na teren skrzyżowania bronią zapory drogowe sterowane przez układ elektroniczny (na podstawie sygnału z sensora). Jako zagrożenie ustalono jednoczesne znalezienie się w rejonie skrzyżowania samochodu i pociągu. Dokładny opis analizowanego systemu zawarto w [GG06].



Rysunek 9.4 Drzewo niezdatności dla przejazdu kolejowego

Drzewo niezdatności stworzone dla tego systemu przedstawiono na rysunku Rysunek 9.4.

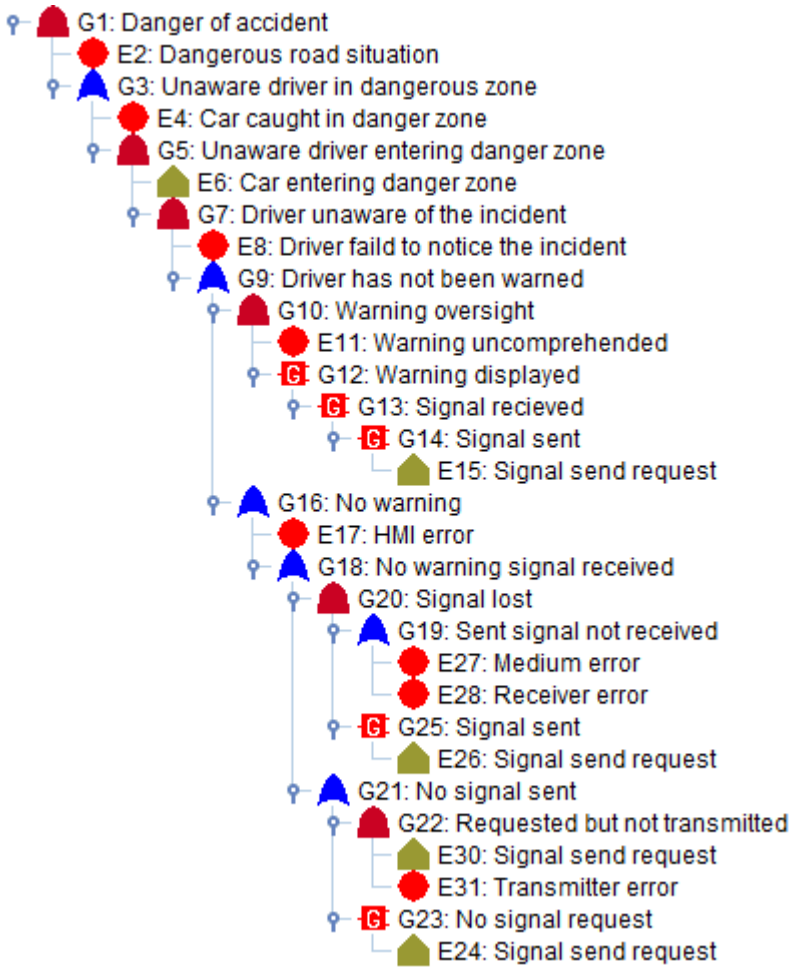
9.1.2.3 System monitoringu ruchu (MR)



Rysunek 9.5 Diagram systemu monitoringu ruchu

Rysunek 9.5 przedstawia schemat systemu monitoringu ruchu mają cyna celu poprawę jego bezpieczeństwa. System ten składa się z centrum monitoringu ruchu, które monitoruje sytuację na drodze i wysyła rekomendacje uczestnikom ruchu odnośnie maksymalnej prędkości oraz minimalnej odległości od poprzedzającego pojazdu. Te rekomendacje dostarczane są do uczestników ruchu poprzez nadajniki komunikacji bezprzewodowej umieszczone wzdłuż trasy. Pojazdy posiadają sensory, które są wykorzystane do komunikowania stanu pojazdu do centrum monitoringu. Dzięki nim, centrum monitoringu jest zdolne do określenia, czy jakiś pojazd nie zatrzymał się na drodze. Jako zagrożenie dla systemu ustalono wjazd pojazdu z kierowcą nieświadomym zagrożenia w rejon zatrzymania się innego pojazdu. Szczegółowy opis analizowanego systemu można znaleźć w [Gol07].

Dla tak wyspecyfikowanego systemu opracowano drzewo niezdatności przedstawione na rysunku Rysunek 9.6:



Rysunek 9.6 Drzewo niezdatności dla systemu monitoringu ruchu

9.1.3 **Wyniki**

Drzewa niezdatności powyżej opisanych systemów zostały poddane badaniom mającym na celu określenie redukcji nakładu pracy wymaganego od analityka dzięki wstępnej selekcji wymagań czasowych oraz redukcji czasu przetwarzania dzięki nowym wersjom algorytmów. Oszacowano również uzyskaną poprawę bezpieczeństwa.

Analiza drzew niezdatności poszczególnych systemów pozwoliła określić zarówno rozmiar problemu (P1), liczbę wyrażeń czasowych, które trzeba przeanalizować celem określenia wymagań (P4), jak i uzyskany wzrost bezpieczeństwa (P5). Liczba wymagań do przeanalizowania opiera się an wynikach działania algorytmu MZPCALC_FULL i redukcji rozwiązania poprzez algorytm REDUCE_MZP. Wyniki zebrano w tabeli Tabela 9.1. W poszczególnych wierszach tabeli są wartości metryk wyszczególnionych w pierwszej kolumnie. Wartości w poszczególnych kolumnach odpowiadają drzewom niezdatności opisanych powyżej systemów (identyfikowanych przez akronim nazwy).

Tabela 9.1 Pomiary dla pytań P1, P4 oraz P5

		PG	PK	MR
Liczba bramek (M1.1):		7	10	17
Liczba zdarzeń prostych (M1.2):	kontrolowalne	2	2	4
	przewidywalne	0	0	1
	obserwowalne	0	1	0
	nieobserwowalne	5	9	8
Liczba wyrażeń czasowych w MZP po redukcji (M4.1):		48	862	221
Liczba wyrażeń czasowych zawartych w kandydujących czasowych wymaganiach bezpieczeństwa wobec systemu (M4.2):		9	28	6
Liczba MZP po redukcji (M5.1)		12	70	13
Liczba unieszkodliwionych MZP (M5.2)		6	12	10

Czasy przetwarzania poszczególnych drzew niezdatności dla różnych algorytmów oraz rozmiar otrzymanego rozwiązania przedstawiono w tabelach Tabela 9.2, Tabela 9.3 oraz Tabela 9.4. Dla każdego pomiaru podano dwie wartości czasu przetwarzania rozdzielone symbolem „/”. Pierwsza stanowi czas przetwarzania drzewa niezdatności dla zadanego algorytmu obliczania MZP oraz algorytmu redukcji nadmiarowości REDUCE_MZP. Druga stanowi czas przetwarzania dla tego samego algorytmu obliczania MZP, ale dla algorytmu redukcji nadmiarowości REDUCE_MZP_ext. Obie te wartości reprezentują wartość średnią uzyskaną na podstawie 20 pomiarów.

Tabela 9.2 Czasy działania poszczególnych algorytmów dla drzewa niezdatności systemu PG

	Analiza wszystkich MZP (MZPCALC_FULL)	Analiza MZP zawierających zdarzenia kontrolowalne (MZPCALC_CONTROLABLE)	Analiza MZP zawierających parę zdarzenie kontrolowalne i obserwowalne (MZPCALC_PAIR)	Analiza MZP nie obejmująca zdarzeń nieobserwowalnych (MZPCALC_PART)
Liczba MZP (M2.1)	24	24	4	4
Liczba wyrażeń w MZP (M.2.2)	99	99	16	16
Czas analizy [ms] (M3.1), w tym	29,0/27,9	27,9/28,5	17,2/17,1	17,5/17,9
Przetwarzanie wstępne [ms]	3,7/3,8	4,0/3,7	3,7/3,5	3,8/4,2
Główny algorytm [ms]	20,4/20,0	19,9/20,6	11,6/11,7	11,8/11,8
Przetwarzanie wyników [ms]	4,9/4,1	4,1/4,2	1,9/1,9	1,9/1,9

Tabela 9.3 Czasy działania poszczególnych algorytmów dla drzewa niezdatności systemu PK

	Analiza wszystkich MZP (MZPCALC_FULL)	Analiza MZP zawierających zdarzenia kontrolowalne (MZPCALC_CONTROLABLE)	Analiza MZP zawierających parę zdarzenie kontrolowalne i obserwowalne (MZPCALC_PAIR)	Analiza MZP nie obejmująca zdarzeń nieobserwowalnych (MZPCALC_PART)
Liczba MZP (M2.1)	196	56	14	14
Liczba wyrażeń w MZP (M.2.2)	2335	718	220	220
Czas analizy [ms] (M3.1), w tym	363,0/354,1	165,9/162,9	91,3/89,9	85,8/83,3
Przetwarzanie wstępne [ms]	4,7/4,9	4,6/4,7	5,1/4,7	5,0/4,8
Główny algorytm [ms]	324,1/316,7	151,5/148,5	80,8/79,9	75,2/73,0
Przetwarzanie wyników[ms]	34,1/32,5	9,8/9,7	5,4/5,3	5,5/5,5

Tabela 9.4 Czasy działania poszczególnych algorytmów dla drzewa niezdatności systemu MR

	Analiza wszystkich MZP (MZPCALC_FULL)	Analiza MZP zawierających zdarzenia kontrolowalne (MZPCALC_CONTROLABLE)	Analiza MZP zawierających parę zdarzenie kontrolowalne i obserwowalne (MZPCALC_PAIR)	Analiza MZP nie obejmująca zdarzeń nieobserwowalnych (MZPCALC_PART)
Liczba MZP (M2.1)	25	20	20	8
Liczba wyrażeń w MZP (M.2.2)	369	327	327	151
Czas analizy [ms] (M3.1), w tym	50,5/49,1	49,2/49,4	48,9/49,1	38,8/38,7
Przetwarzanie wstępne [ms]	6,2/5,7	6,1/5,7	5,7/5,6	5,9/6,2
Główny algorytm[ms]	35,2/34,5	34,3/35,0	34,6/34,7	25,8/26,5
Przetwarzanie wyników [ms]	9,2/8,9	8,7/8,7	8,6/8,7	7,0/6,0

9.1.4 **Analiza wyników**

Osiągnięte wyniki potwierdzają znaczne zmniejszenie pracochłonności określania wymagań bezpieczeństwa wobec analizowanego systemu. Osiągnięta redukcja liczby koniecznych do przeanalizowania wyrażeń wyniosła dla poszczególnych systemów:

- system palnika gazowego - 81%,
- przejazd kolejowy - 97%,
- system monitoringu ruchu - 97%.

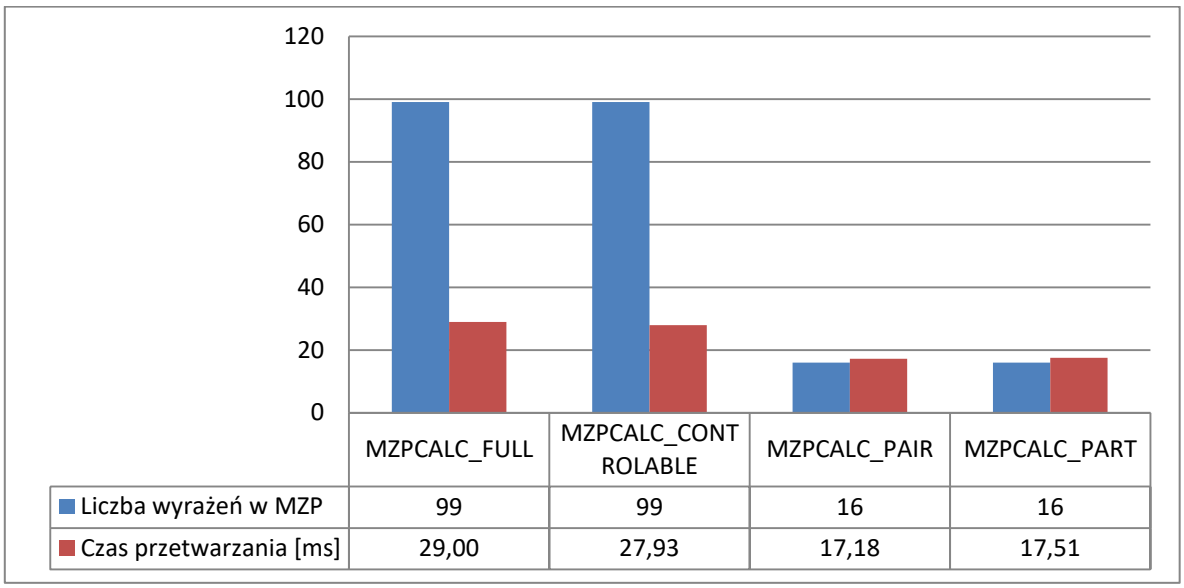
Jednocześnie uzyskany potencjalny wzrost bezpieczeństwa też był wyskoki, choć wystąpiły tu duże wahania. Dla poszczególnych systemów odsetek scenariuszy hazardów, których można uniknąć poprzez wprowadzenie czasowych wymagań bezpieczeństwa, wyniósł:

- system palnika gazowego - 50%,
- przejazd kolejowy - 17%,
- system monitoringu ruchu - 77%.

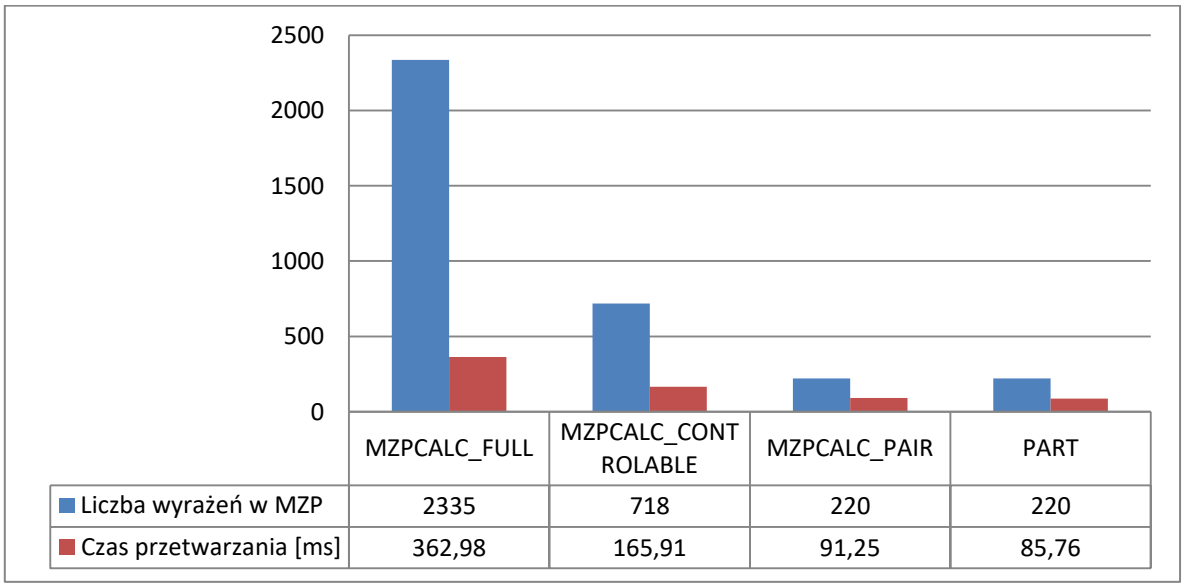
Nie zaobserwowano zależności pomiędzy osiągniętą redukcją liczby wyrażeń wymagających analizy a rozmiarem zadania wyrażonego liczbą bramek bądź liczbą poszczególnych rodzajów zdarzeń prostych. Redukcja ta nie zależała również od osiągniętego wzrostu bezpieczeństwa.

Pomiędzy czasem działania algorytmów redukcji nadmiarowości REDUCE_MZP oraz REDUCE_MZP_ext nie występują znaczące różnice. Przewagi można doszukiwać się po stronie REDUCE_MZP_ext, ale jest ona drobna.

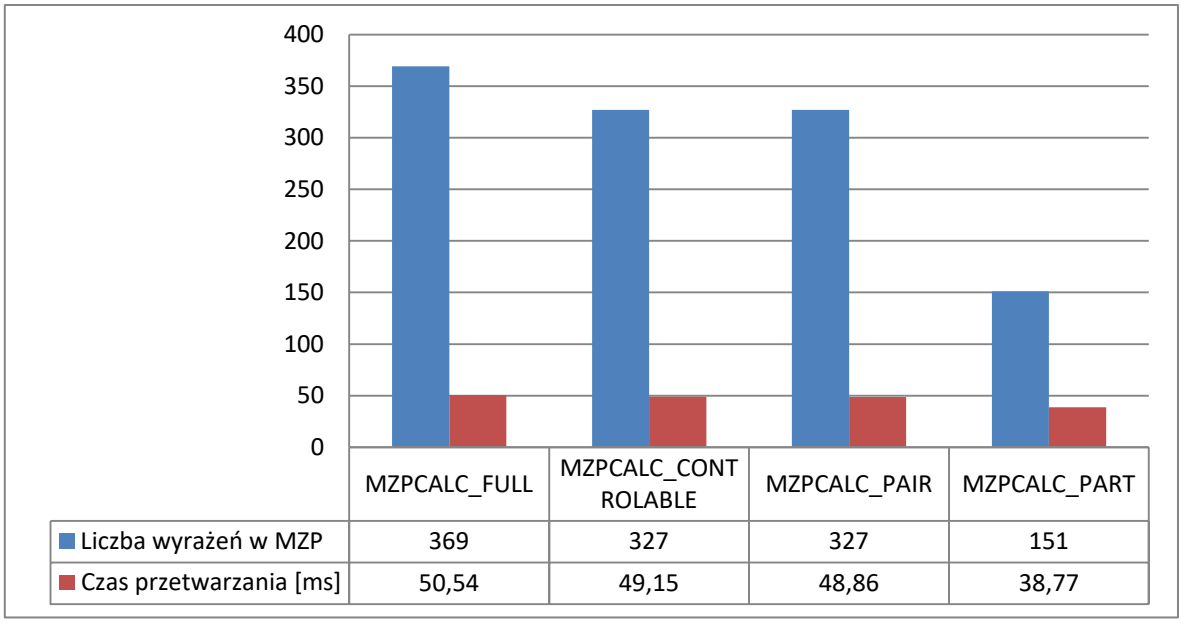
Redukcję czasu przetwarzania dla poszczególnych algorytmów MZPCALC przedstawiono na wykresach poniżej (Rysunek 9.7, Rysunek 9.8, Rysunek 9.9).



Rysunek 9.7 Czasy wykonywania algorytmów na drzewie niezdatności systemu PG



Rysunek 9.8 Czasy wykonywania algorytmów na drzewie niezdatności systemu PK



	MZPCALC_FULL	MZPCALC_CONTROLABLE	MZPCALC_PAIR	MZPCALC_PART
Liczba wyrażeń w MZP	369	327	327	151
Czas przetwarzania [ms]	50,54	49,15	48,86	38,77

Rysunek 9.9 Czasy wykonywania algorytmów na drzewie niezdatności systemu MR

Wyniki pomiaru czasów analizy potwierdzają poczynione założenia. Redukcja czasu analizy miała nastąpić poprzez redukcję rozmiaru rozwiązania i taka zależność została zaobserwowana.

W wynikach można zauważyć, iż stopień redukcji czasu analizy nie był współmierny do redukcji rozmiaru rozwiązania. Ma to dwie przyczyny:

- niezależnie od rozkładu poszczególnych typów zdarzeń, wszystkie bramki są zawsze przetwarzane,
- redukcja budowanego MZP następuje dopiero w momencie, gdy wiadomo, że nie zostanie z niej wygenerowane MZP zawierające poszukiwane zdarzenia. Może to nastąpić gdy przetwarzane są dość niskie bramki drzewa niezdatności.

Ponadto w jednym przypadku można zauważyć redukcję czasu przetwarzania wykraczającą poza typowe fluktuacje bez zmiany rozmiaru rozwiązania. Dotyczy to algorytmów MZPCALC_PAIR i MZPCALC_PART dla drzewa niezdatności systemu PK (patrz Rysunek 9.8). Sytuacja taka jest możliwa, ponieważ algorytm MZPCALC_PART mógł odfiltrować odrzucone MZP we wcześniejszym etapie przetwarzania niż MZPCALC_PAIR. Algorytm MZPCALC_PART wymaga jednak bardziej skomplikowanego przetwarzania, a nie można zagwarantować wystąpienia korzyści z jego zastosowania w każdym przypadku. Dlatego nie należy formułować ogólnych zależności na podstawie tego jednostkowego przypadku.

Czasy przetwarzania wstępnego nieco się od siebie różnią, chociaż teoretycznie dla poszczególnych drzew niezdatności powinny być niezmienne. Również czas pracy algorytmów MZPCALC dla wybranego drzewa niezdatności i wersji algorytmu nie są idealnie zbieżne (patrz wiersz Główny algorytm). Różnicę tą przypisuje się wpływowi innych programów pracujących równolegle z narzędziem (np. system operacyjny).

9.2 CS-2 Drugie studium przypadku

9.2.1 Cel studium przypadku

Drugie studium przypadku ma na celu eksperymentalne oszacowanie złożoności obliczeniowej.

Cel i zakres studium opisano za pomocą szablonu GQM:

Zbadać metodę TREM w celu oceny wpływu rozmiaru problemu na czas przetwarzania.

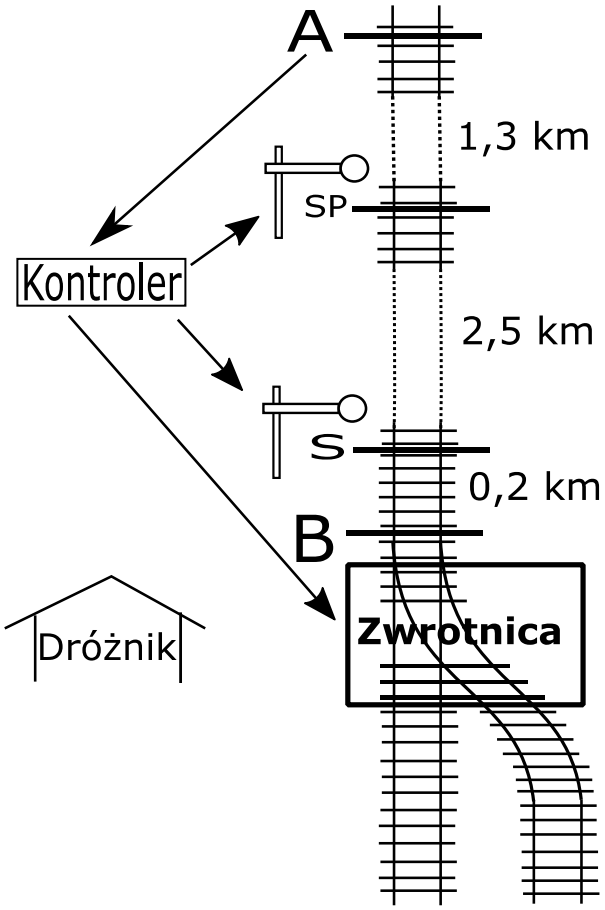
Pytania i metryki:

- P1: Jaki jest rozmiar problemu (rozmiar analizowanych drzew niezdatności)?
    - M1.1: Liczba bramek w drzewie niezdatności.
    - M1.2: Liczba zdarzeń prostych, z podziałem na typy.
- P2: Jaki jest rozmiar rozwiązania problemu?
    - M2.1: Liczba Zbiorów Przyczyn (ZP; przed redukcją)[13].
    - M2.2: Liczba wyrażeń czasowych zawartych w Zbiorach Przyczyn (przed redukcją).
- P3: Jaki jest koszt obliczeń?
    - M3.1 Czas analizy problemu (dla poszczególnych wariantów algorytmu).

9.2.2 Przedmiot badań

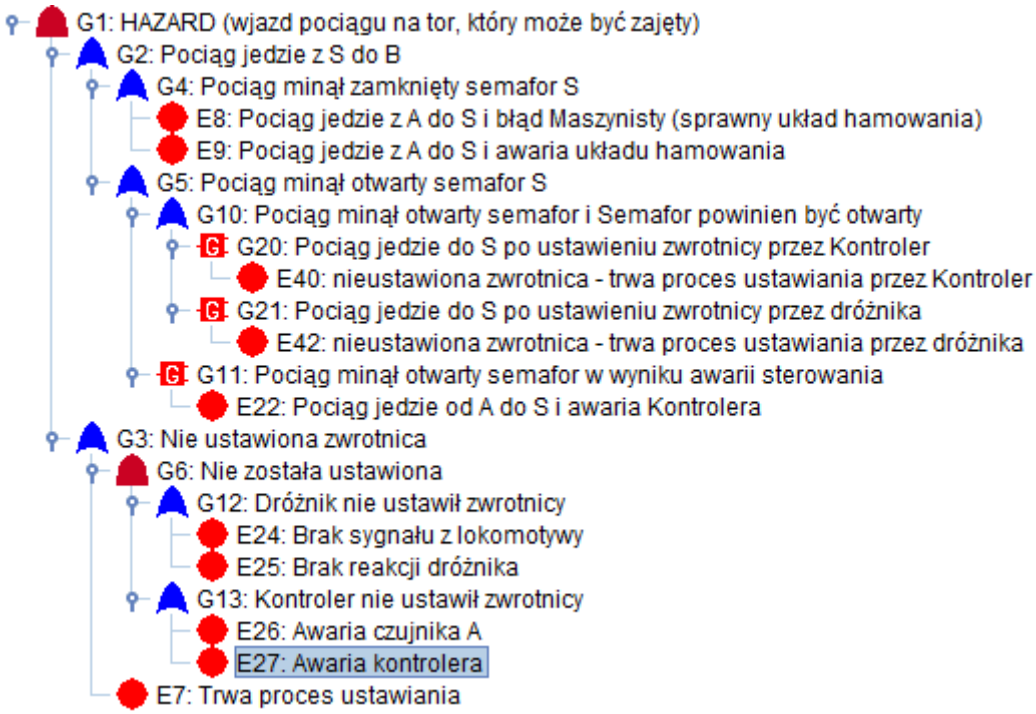
Analizowany w tym studium przypadku system to rozjazd kolejowy, opis którego został zaczerpnięty z pracy [Skr05]. Składa się on z toru, który przy wjeździe na stację kolejową rozdziela się na dwa. Rozjazd jest obsługiwany przez zwrotnicę, wjazdu do której broni semafor. Semafor (oraz semafor pomocniczy) obsługiwane są automatycznie przez system kontrolera. Dodatkowo rozjazd nadzorowany jest przez dróżnika, który w razie potrzeby ręcznie zmienia ustawienia semaforów i zwrotnicy. Szczegółowy opis analizowanego systemu można znaleźć w [Skr05]. Na Rysunek 9.10 został przedstawiony schemat tego systemu.

[13] W tym studium przypadku analizowane były duże drzewa niezdatności, w związku z czym zliczanie liczby MZP, a w szczególności liczby wyrażeń czasowych w MZP przez ręczną analizę rozpatrywanych drzew niezdatności byłoby bardzo podatne na błędy. W związku z czym zdecydowano się na przeprowadzanie zliczania w narzędziu wspomagającym, co jednak wiązało się z koniecznością przeprowadzenia obliczeń przed zastosowaniem algorytmu REDUCE_MZP/REDUCE_MZP_ext



Rysunek 9.10 Schemat rozjazdu (za [Skr05])

W pracy [SKR05] opracowano dla tego systemu drzewo niezdatności. Dla oceny złożoności omawianego w niej algorytmu (INES) dokonano pomiaru czasu analizy tego drzewa, oraz drzewa zbudowanego z wielu jego kopii. Dla oceny złożoności czasowej algorytmu *TREM* dokonano podobnego zabiegu. Drzewo stworzone w przytoczonej pracy, jak również drzewa stworzone z jego powtórzeń zostały przedstawione w narzędziu PolymorphFT2 (patrz Rysunek 9.11). Dodatkowo poszczególnym zdarzeniom prostym została arbitralnie przypisana obserwowalność (nie uwzględniając konstrukcji systemu wejściowego).



Rysunek 9.11 Drzewo niezdatności dla systemu rozjazdu kolejowego

Omawiane studium przypadku zostało również w skrótowej postaci przedstawione w [Gol13].

9.2.3 **Wyniki**

Parametry czasowe poszczególnych wariantów algorytmu były mierzone w narzędziu analitycznym poprzez pomiar czasu systemowego (wyrażonego w nanosekundach) w wybranych miejscach programu. Uzyskane w ten sposób wartości zostały przez program wypisane do pliku tekstowego, po czym podlegały dalszej obróbce.

Uzyskane wyniki, będące średnią z 10 powtórzeń, są przedstawione w Tabela 9.5, Tabela 9.6, Tabela 9.7 oraz Tabela 9.8. W nagłówkach tabel podane są mierzone wielkości, w poszczególnych wierszach są wartości pomiarów dla poszczególnych drzew niezdatności identyfikowanych przez numer podany w pierwszej kolumnie. W poszczególnych tabelach drzewa identyfikowane przez ten sam numer są identyczne, ponadto dla zwiększenia czytelności wyniki pomiarów dla drzewa numer 1 zostały powtórzone pod numerem 6. Dla pomiarów czasowych podane są dwie wartości rozdzielone znakiem „/”, które obrazują czas analizy przy zastosowaniu algorytmu redukcji nadmiarowości odpowiednio REDUCE_MZP oraz REDUCE_MZP_ext.

Tabela 9.5 Analiza poszczególnych drzew niezdatności algorytmem MZPCALC_FULL

Numer	Liczba bramek (M1.1)	Liczba zdarzeń (M1.2)				Liczba ZP (M2.1)	Liczba wyrażeń czasowych w ZP (M2.2)	Czas analizy [ms] (M3.1)			
		Kontrolowalne	Przewidywalne	Obserwowalne	Nieobserwowalne			Całkowity	Przetwarzanie wstępne	Główny algorytm	Przetwarzanie wyników
1	12	1	0	1	8	170	2010	388,5/391,9	5,2/5,7	329,7/331,8	53,6/54,4
2	24	1	0	1	17	380	4580	1006,5/1043,8	7,1/7,7	866,2/897,1	133,2/139,1
3	36	1	0	1	26	1900	28308	5153,0/5397,1	9,2/9,5	4013,9/3977,1	1129,9/1410,5
4	48	1	0	1	36	3572	62980	13085,7/13572,4	12,7/11,4	9545,2/9329,6	3527,8/4231,4
5	60	1	0	1	45	5092	86708	18537,7/19631,7	14,0/13,5	12610,3/12675,9	5913,4/6942,4
6	12	1	0	1	8	170	2010	388,5/391,9	5,2/5,7	329,7/331,8	53,6/54,4
7	24	2	0	2	15	380	4580	998,0/1036,0	7,3/7,1	846,9/888,8	143,9/140,1
8	36	3	0	3	22	1900	28308	5106,3,6/5257,9	9,4/10,5	3959,0/3963,1	1137,9/1284,3
9	48	4	0	4	30	3572	62980	13269,7/13490,3	12,6/11,9	9742,6/9367,8	3514,6/4110,6
10	60	5	0	5	37	5092	86708	18503,3/19931,1	14,2/13,6	12560,3/13023,8	5928,8/6893,7
11	12	2	1	1	6	170	2010	394,9/385,4	5,6/5,2	331,1/322,1	58,2/58,1
12	24	4	2	2	11	380	4580	1024,2/1036,5	8,0/7,2	866,2/876,0	150,1/153,3
13	36	6	3	3	16	1900	28308	5272,9/5414,9	8,9/9,7	4061,5/4001,2	1202,5/1404,0
14	48	8	4	4	22	3572	62980	13273,9/13479,3	12,5/12,9	9495,6/9221,0	3765,8/4245,3
15	60	10	5	5	27	5092	86708	18945,9/19615,5	15,3/14,0	12823,4/12507,4	6107,2/7094,1

Tabela 9.6 Analiza poszczególnych drzew niezdatności algorytmem MZPCALC_CONTROLABLE

Numer	Liczba bramek (M1.1)	Liczba zdarzeń (M1.2)				Liczba ZP (M2.1)	Liczba wyrażeń czasowych w ZP (M2.2)	Czas analizy [ms] (M3.1)			
		Kontrolowalne	Przewidywalne	Obserwowalne	Nieobserwowalne			Całkowity	Przetwarzanie wstępne	Główny algorytm	Przetwarzanie wyników
1	12	1	0	1	8	34	314	68,7/69,5	4,9/5,0	56,2/56,7	7,7/7,8
2	24	1	0	1	17	76	702	169,0/168,3	7,6/6,9	142,4/145,1	19,0/16,3
3	36	1	0	1	26	76	702	244,8/241,4	9,8/8,9	219,3/216,8	15,7/15,7
4	48	1	0	1	35	76	702	251,1/254,6	10,4/9,9	225,1/228,8	15,7/15,9
5	60	1	0	1	44	76	702	261,5/260,6	10,7/10,7	235,1/234,1	15,7/15,8
6	12	1	0	1	8	34	314	68,7/69,5	4,9/5,0	56,2/56,7	7,7/7,8
7	24	2	0	2	15	84	728	194,9/199,9	6,9/7,1	172,0/176,5	16,0/16,4
8	36	3	0	3	22	198	1670	461,2/458,4	9,0/8,8	414,3/411,0	37,9/38,6
9	48	4	0	4	29	316	2710	974,6/962,2	10,7/10,2	880,6/866,7	83,3/85,3
10	60	5	0	5	36	430	4094	1344,5/1367,0	12,0/11,2	1208,4/1227,8	124,2/128,0
11	12	2	1	1	6	98	1228	298,7/306,7	5,0/4,6	255,7/263,7	38,0/38,4
12	24	4	2	2	11	212	2794	814,5/797,2	7,6/7,4	710,9/690,6	96,0/99,2
13	36	6	3	3	16	1270	19896	4173,3/4397,2	9,3/9,0	3413,4/3568,9	750,5/819,4
14	48	8	4	4	21	2396	44526	10247,4/10538,3	11,8/11,7	7971,4/8022,2	2264,2/2504,5
15	60	10	5	5	26	3454	61628	14356,0/14774,7	13,7/13,5	10760,1/10810,2	3582,1/3951,0

Tabela 9.7 Analiza poszczególnych drzew niezdatności algorytmem MZPCALC_PAIR

Numer	Liczba bramek (M1.1)	Liczba zdarzeń (M1.2)				Liczba ZP (M2.1)	Liczba wyrażeń czasowych w ZP (M2.2)	Czas analizy [ms] (M3.1)			
		Kontrolowalne	Przewidywalne	Obserwowalne	Nieobserwowalne			Całkowity	Przetwarzanie wstępne	Główny algorytm	Przetwarzanie wyników
1	12	1	0	1	8	16	152	52,6/52,9	4,8/5,3	43,1/43,0	4,7/44,6
2	24	1	0	1	17	16	176	83,2/81,2	7,3/7,1	70,3/68,5	5,7/5,7
3	36	1	0	1	26	16	176	99,0/100,9	9,1/9,2	84,3/86,1	5,6/5,6
4	48	1	0	1	35	0	0	34,9/35,2	9,7/10,1	25,2/25,0	0,0/0,0
5	60	1	0	1	44	0	0	40,3/41,1	10,7/11,0	29,6/30,1	0,0/0,0
6	12	1	0	1	8	16	152	52,6/52,9	4,8/5,3	43,1/43,0	4,7/44,6
7	24	2	0	2	15	34	332	135,7/133,8	6,7/7,2	120,7/118,4	8,4/8,2
8	36	3	0	3	22	84	776	308,7/303,4	9,1/9,5	282,4/276,7	17,2/17,3
9	48	4	0	4	29	134	1260	694,6/666,5	10,6/10,1	655,6/628,7	28,4/27,7
10	60	5	0	5	36	184	1908	952,1/934,3	11,2/10,7	887,8/870,9	53,1/52,6
11	12	2	1	1	6	64	816	274,6/274,3	4,8/5,0	243,0/242,6	26,9/26,7
12	24	4	2	2	11	134	1842	755,1/747,9	7,7/7,4	683,6/677,4	63,8/63,1
13	36	6	3	3	16	964	15558	3979,6/3983,8	9,0/10,4	3387,6/3358,8	583,0/614,5
14	48	8	4	4	21	1826	35104	9713,1/9670,7	11,1/11,8	8023,8/7825,8	1678,2/1833,1
15	60	10	5	5	26	2656	48820	13208,9/13409,1	13,7/14,2	10481,3/10544,8	2713,9/2850,0

Tabela 9.8 Analiza poszczególnych drzew niezdatności algorytmem MZPCALC_PART

Numer	Liczba bramek (M1.1)	Liczba zdarzeń (M1.2)				Liczba ZP (M2.1)	Liczba wyrażeń czasowych w ZP (M2.2)	Czas analizy [ms] (M3.1)			
		Kontrolowalne	Przewidywalne	Obserwowalne	Nieobserwowalne			Całkowity	Przetwarzanie wstępne	Główny algorytm	Przetwarzanie wyników
1	12	1	0	1	8	16	152	53,0/53,3	5,4/5,1	43,1/43,5	4,6/4,7
2	24	1	0	1	17	16	176	80,6/81,4	7,2/7,2	67,8/68,5	5,6/5,6
3	36	1	0	1	26	16	176	99,9/101,1	9,3/9,5	85,0/86,0	5,6/5,6
4	48	1	0	1	35	0	0	35,6/35,7	10,7/10,1	24,9/25,6	0,0/0,0
5	60	1	0	1	44	0	0	40,8/40,7	10,4/10,2	30,3/30,5	0,0/0,0
6	12	1	0	1	8	16	152	53,0/53,3	5,4/5,1	43,1/43,5	4,6/4,7
7	24	2	0	2	15	34	332	133,2/133,0	7,8/7,3	117,1/117,3	8,3/8,3
8	36	3	0	3	22	68	664	293,4/281,6	9,0/9,2	269,0/257,0	15,4/15,4
9	48	4	0	4	29	110	1084	602,8/595,4	11,4/10,9	567,0/559,8	24,5/24,6
10	60	5	0	5	36	144	1620	842,0/837,5	11,0/11,3	787,0/781,8	44,0/44,4
11	12	2	1	1	6	60	704	236,0/234,7	4,9/4,9	207,2/214,2	23,9/15,6
12	24	4	2	2	11	126	1606	627,4/620,1	7,6/7,2	562,8/574,1	57, 1/38,7
13	36	6	3	3	16	956	13530	3649,2/3512,6	9,0/9,8	3133,5/3130,3	506,7/372,6
14	48	8	4	4	21	1806	31636	8943,2/8434,5	11,6/12,1	7378,7/7375,7	1552,9/1046,7
15	60	10	5	5	26	2628	43508	12446,8/11458,4	13,4/13,3	9803,1/9702,2	2630,4/1742,9

9.2.1 **Analiza wyników**

Przed przystąpieniem do omówienia wyników analizowane drzewa niezdatności zostaną podzielone na trzy ciągi:

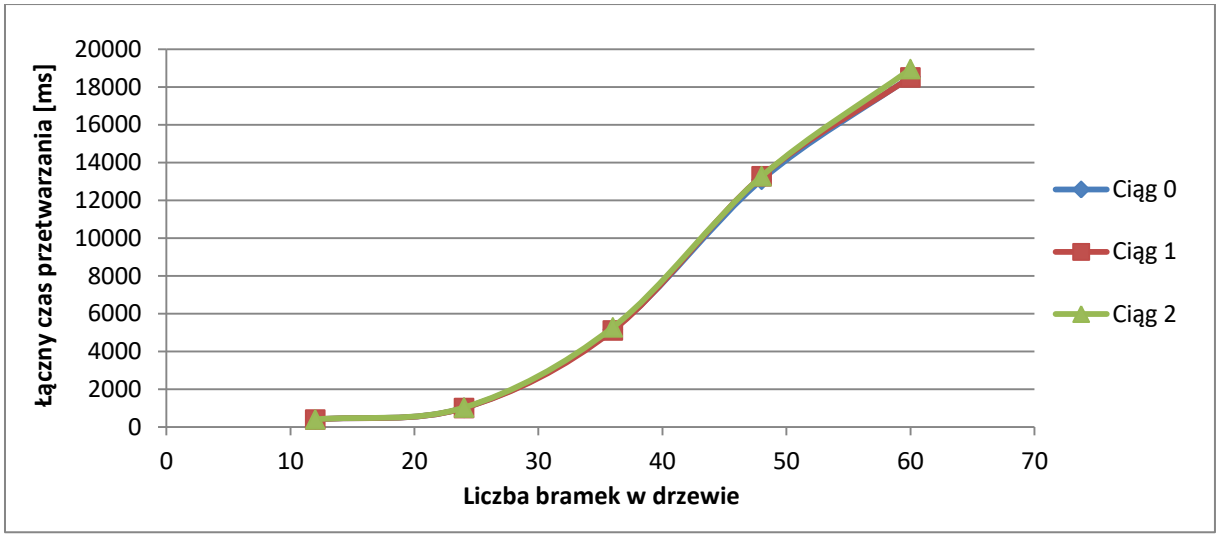
- Ciąg 0: drzewa o numerach 1-5. Wraz ze wzrostem rozmiaru drzewa rośnie tylko liczba zdarzeń nieobserwowalnych.

- Ciąg 1: drzewa o numerach 6-10.Wraz z rozmiarem drzewa rośnie liczba zdarzeń kontrolowalnych oraz obserwowalnych.
- Ciąg 2: drzewa o numerach 11-15. Większa niż w ciągu 1 liczba zdarzeń innych niż nieobserwowalne. Wraz z rozmiarem drzewa rośnie liczba zdarzeń kontrolowalnych, przewidywalnych oraz obserwowalnych.

Innymi słowy w ciągu 0 przy powiększeniu drzewa o nowe bramki i zdarzenia liczba zdarzeń kontrolowalnych i obserwowalnych nie ulega zwiększeniu. W ciągach 1 oraz 2 liczba zdarzeń innych niż nieobserwowalne rośnie proporcjonalnie do rozmiaru drzewa, tyle że w ciągu 2 ich liczba w drzewie początkowym jest dwukrotnie większa niż w ciągu 1.

Dodatkowo należy zaznaczyć, że w ciągi te zostały tak dobrane, by drzewa na tej samej pozycji w ciągu miały tę samą liczbę bramek oraz zdarzeń prostych (różni je tylko klasyfikacja poszczególnych zdarzeń).

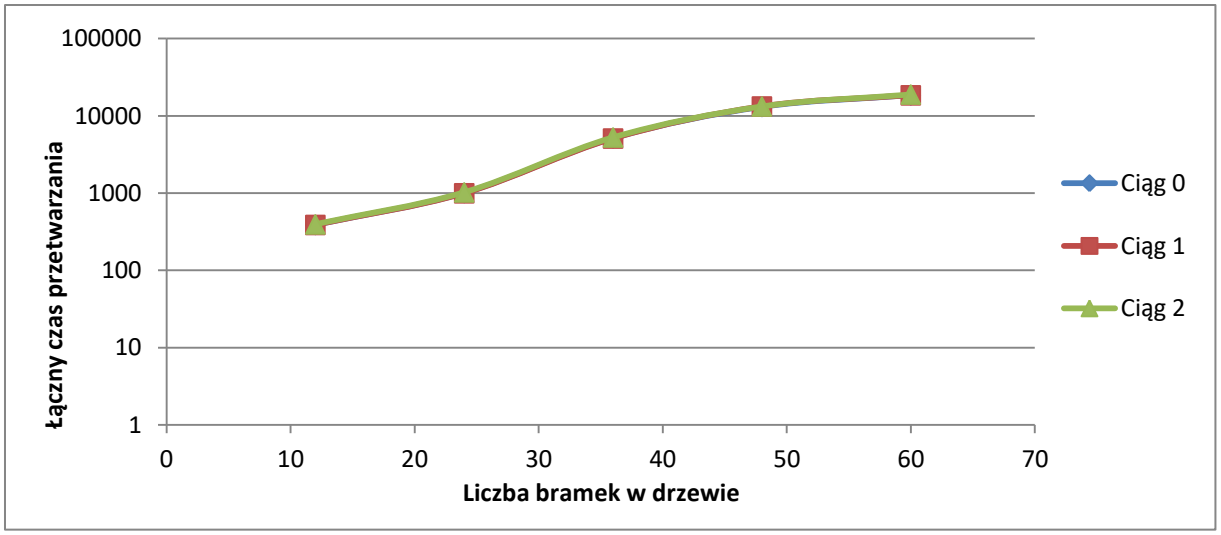
Wyniki dla algorytmu MZPCALC_FULL przedstawiają się następująco:



Rysunek 9.12 Łączny czas przetwarzania dla algorytmu MZPCALC_FULL

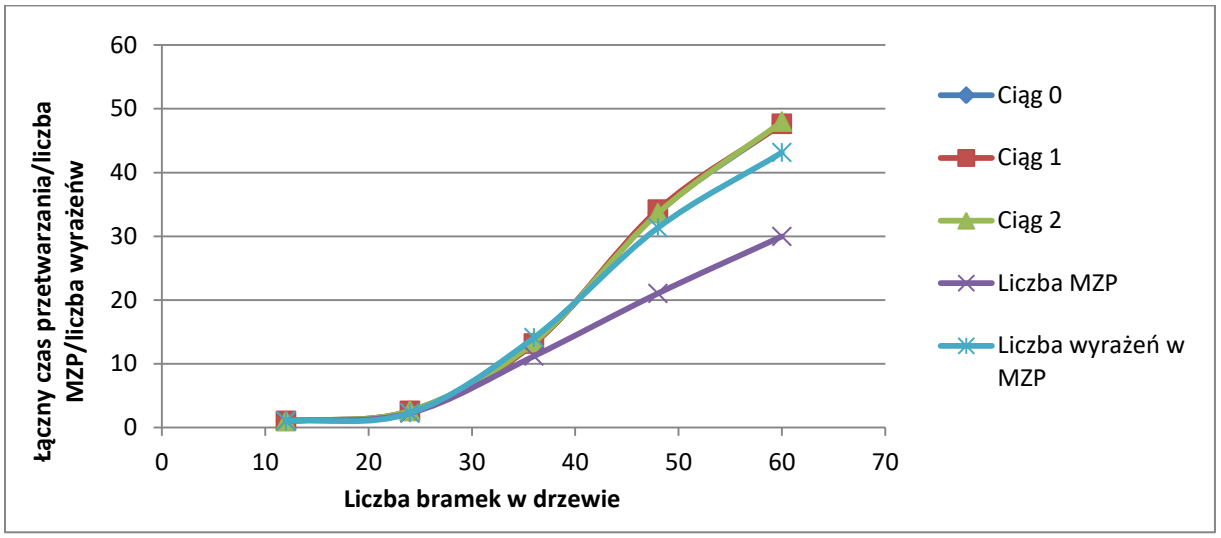
Jak widać na powyższym wykresie czas przetwarzania jest bardzo zbliżony dla wszystkich ciągów. Jest to oczekiwane zachowanie, gdyż badane ciągi różniły się między sobą (na poszczególnych pozycjach) tylko klasyfikacją zdarzeń prostych, a ta nie ma wpływu na algorytm MZPCALC_FULL. Klasyfikacja zdarzeń w tym przypadku ma wpływ jedynie na czas przetwarzania algorytmu REQCALC.

Jednocześnie da się zaobserwować, że czas przetwarzania znacząco wzrasta wraz ze wzrostem rozmiaru drzewa przy trzecich i czwartych elementach ciągu, przy piątym zaś ten wzrost trochę zwalnia. Nie należy się jednak spodziewać złożoności liniowej, gdyż wraz z wzrostem liczby bramek rośnie zarówno liczba grafów do przeanalizowania, jak również rozmiary poszczególnych grafów. Sama zaś złożoność przetwarzania poszczególnych grafów jest sześcienna. Przedstawiając jednak te same dane na wykresie, którego oś rzędnych jest w skali logarytmicznej (Rysunek 9.13), widać że przedstawiona funkcja jest wypukła. Umożliwia to oszacowanie złożoności wykorzystanych algorytmów jako wielomianowej.



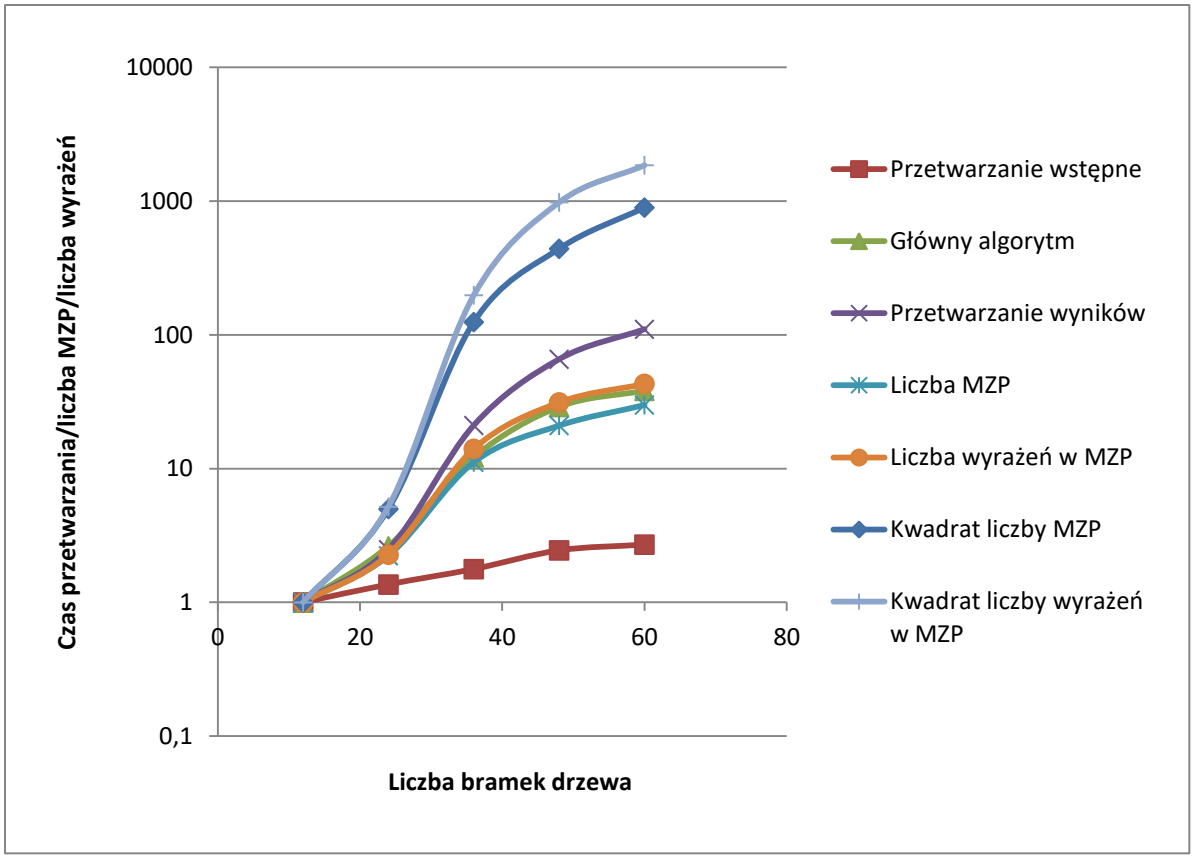
Rysunek 9.13 Łączny czas przetwarzania dla algorytmu MZPCALC_FULL (oś rzędnych w skali logarytmicznej)

Aby zobrazować zależność czasu przetwarzania drzew niezdatności od rozmiaru wyniku analizy do poniższego wykresu dodano liczbę MZP oraz łączną liczbę wyrażeń w MZP, a wielkości przeskalowano tak, aby dla drzew o liczbie bramek 12 wszystkie miały wartość 1.



Rysunek 9.14 Czas przetwarzania dla algorytmu MZPCALC_FULL zestawiony z rozmiarem wyniku

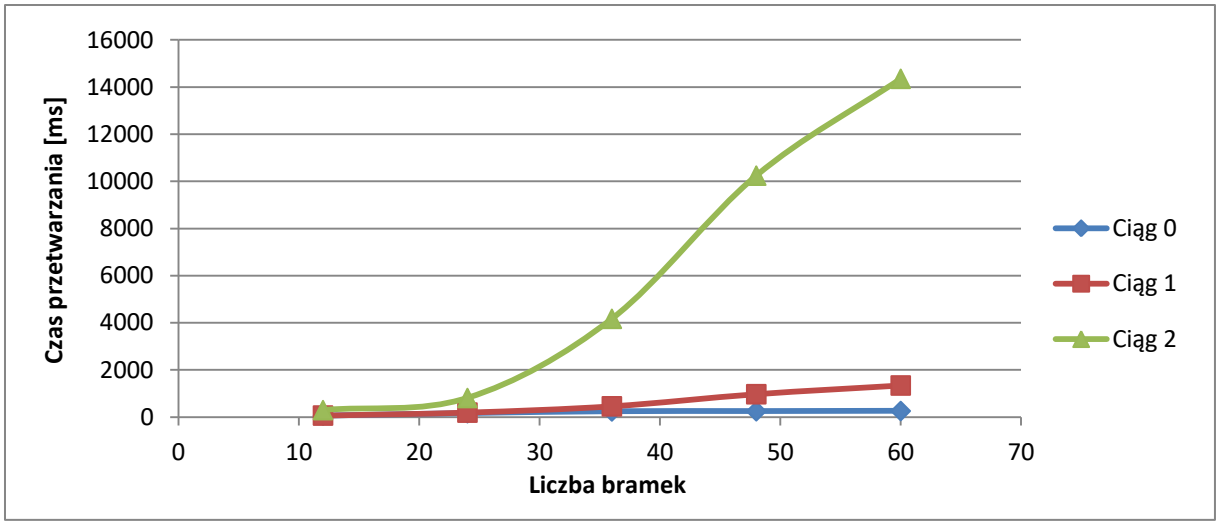
Z powyższego wykresu można wnioskować, że czas przetwarzania drzewa jest proporcjonalny do rozmiaru rozwiązania (wyrażonego łączną liczbą wyrażeń we wszystkich zbiorach przyczyn). Dlatego na następnym wykresie umieszczono czasy przetwarzań dla poszczególnych etapów algorytmu dla Ciągu 0. Dla większej czytelności oś rzędnych przedstawiono w skali logarytmicznej.



Rysunek 9.15 Czasy wykonywania poszczególnych etapów analizy dla Ciągu 0

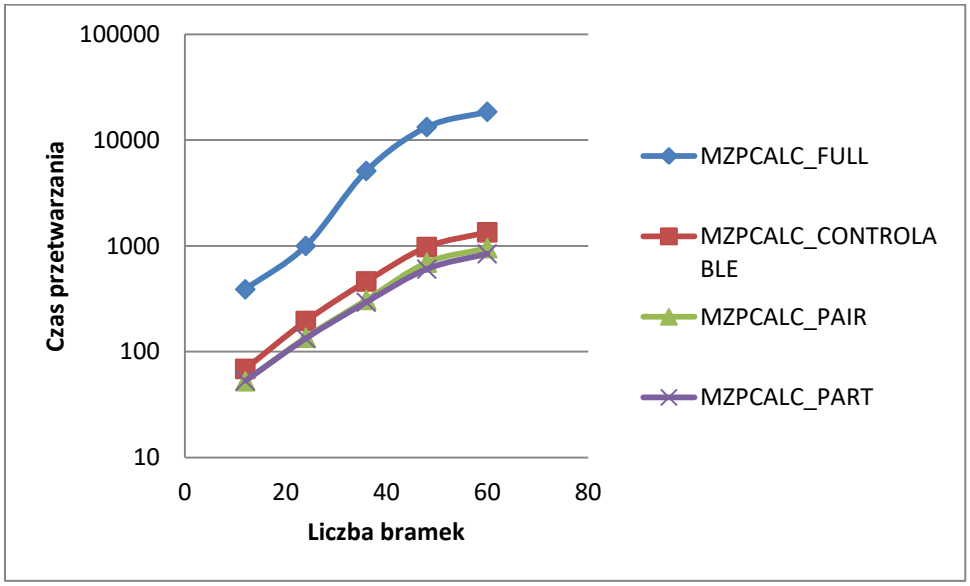
Z powyższego wykresu widać, że czas przetwarzania wstępnego rośnie wolniej niż liczba MZP w rozwiązaniu i jest w przybliżeniu liniowy względem liczby bramek. Czas obliczania MZP jest ograniczony z dołu liczbą MZP, a z góry łączną liczbą wyrażeń w nich zawartą. Najgorszą złożoność czasową ma przetwarzanie wyników. Czas przetwarzania wyników rośnie szybciej niż łączna liczba wyrażeń w MZP ale wolniej niż kwadrat liczby MZP (oraz kwadrat łącznej liczby wyrażeń zawartych w MZP). Jest to spowodowane koniecznością porównania ze sobą wszystkich par MZP celem redukcji zbiorów nieminimalnych. Powyższe zależności są również prawdziwe dla pozostałych ciągów.

Czasy przetwarzania drzewa z użyciem algorytmu analizującego MZP zawierające zdarzenie kontrolowalne (MZPCALC_CONTROLABLE) powinny się od siebie różnić (najniższe dla Ciągu 0, najwyższe dla Ciągu 2). Potwierdzają to pomiary, co zobrazowano na poniższym wykresie.



Rysunek 9.16 Czas przetwarzania dla algorytmu MZPCALC_CONTROLABLE

Wykres ten pokazuje również jak istotną rolę dla czasu przetwarzania ma ilość zdarzeń kontrolowalnych w drzewie: ostatnie drzewo z Ciągu 2 było przetwarzane ponad 10 razy dłużej niż drzewo o tej samej liczbie bramek z Ciągu 1. Również dla algorytmów MZPCALC_PAIR oraz MZPCALC_PART zachowana została zależność, że najszybciej przetwarzany jest Ciąg 0, a najwolniej Ciąg 2.



Rysunek 9.17 Czas przetwarzania Ciągu 1 dla poszczególnych wersji algorytmu

Rysunek 9.17 przedstawia zależność pomiędzy czasem przetwarzania drzew Ciągu 1 a wybranym algorytmem. Z wykresu tego łatwo odczytać, że im bardziej wybrany algorytm ogranicza rozmiar rozwiązania tym szybciej się wykonuje. Jeśli jednak uszeregować algorytmy względem skali ograniczenia rozmiaru rozwiązania i porównać z poprzednim osiągniętą redukcję czasu przetwarzania, to największą różnicę otrzymano dla algorytmu MZPCALC_CONTROLABLE względem MZPCALC_FULL. Pozostałe dwa algorytmy dalej zwiększają te oszczędności, ale uzysk już nie jest tak duży. Również dla pozostałych ciągów zależności zaobserwowane na Rysunek 9.17 są zachowane,

chociaż różnice pomiędzy czasami przetwarzania dla poszczególnych algorytmów są większe dla Ciągu 0, a mniejsze dla Ciągu 2.

Należy również zauważyć, że w przeciwieństwie do poprzednich eksperymentów czas przetwarzania wyników jest mniejszy dla algorytmu redukcji nadmiarowości REDUCE_MZP w porównaniu z REDUCE_MZP_ext dla wszystkich algorytmów przetwarzania poza MZPCALC_PART, w szczególności dla MZPCALC_FULL.

9.3 Wnioski z badań walidacyjnych

Pierwsze z przeprowadzonych studiów przypadku potwierdziło znaczną redukcję wyrażeń czasowych, które musiałby przeanalizować analityk celem określenia wymagań czasowych wobec systemu. Skala redukcji wynosiła od 81% do 97%.

W powyższym omówieniu celowo nie podano średniego stopnia redukcji nakładu pracy analityka, gdyż uznano, że liczba przeanalizowanych drzew niezdatności nie uprawnia do wyciągania wniosków o charakterze statystycznym. Jednocześnie oczekuje się, że ponieważ we wszystkich przeanalizowanych przypadkach uzyskano ogromną redukcję, w zastosowaniu do dowolnego innego drzewa niezdatności również będzie ona znacząca.

Przeprowadzone w CS-2 (rozdział 9.2) eksperymenty umożliwiają oszacowanie złożoności czasowej analizy drzew niezdatności jako wielomianowej względem liczby bramek drzewa.

Jednocześnie w obu studiach przypadków udało się uzyskać redukcję czasu przetwarzania przez redukcję rozmiaru wyników. Redukcja ta występowała w przypadku każdego algorytmu względem jego poprzednika.

A więc uzyskano redukcję czasów przetwarzania drzewa poprzez zmianę algorytmu:

- z MZPCALC_FULL na MZPCALC_CONTROLABLE,
- z MZPCALC_CONTROLABLE na MZPCALC_PAIR,
- z MZPCALC_PAIR na MZPCALC_PART.

Należy zauważyć, że w każdym z powyższych przypadków redukcja czasu przetwarzania warunkowana była zmniejszeniem rozmiaru danych wynikowych. W drugim studium przypadku następowało to prawie w każdym przypadku (z wyjątkiem przetwarzania dużych drzew Ciągu 0 algorytmami MZPCALC_PAIR i MZPCALC_PART), w pierwszym jednak rzadziej. W każdym jednak przypadku czas przetwarzania się nie zwiększał.

Odnośne złożoności poszczególnych części algorytmu, uzyskane wyniki pokazują, że wzrost czasu przetwarzania algorytmu MZPCALC_FULL jest z dwóch stron ograniczony wzrostem rozmiaru rozwiązania liczonym jako liczba MZP od dołu, a jako łączna liczba wyrażeń w MZP od góry. Pokazuje to, że ograniczanie rozmiaru rozwiązania jest właściwym podejściem do ograniczania czasowej złożoności algorytmu.

Wyniki pokazały również, że względnie szybciej, niż rozmiar wyniku, rósł czas jego przetwarzania. Nigdy nie doszło do sytuacji, aby było on większy niż czas pracy głównego algorytmu. Niemniej, gdyby

dla większych drzew czas ten zacząłby stanowić znaczący problem warto zauważyć, że stosowanie algorytmów redukujących rozmiar rozwiązania redukuje również ten problem.

Porównując czas pracy algorytmów REDUCE_MZP i REDUCE_MZP_ext da się zauważyć, że choć występują przypadki, w których algorytm REDUCE_MZP_ext wykazuje przewagę, to w przypadku drzew najbardziej wymagających obliczeniowo przewaga jest po stronie REDUCE_MZP. (Przez drzewa najbardziej wymagające obliczeniowo rozumie się drzewa największe z najmniej zredukowanymi zbiorami MZP przez algorytm główny {MZPCALC_*}).

Podsumowując, studia przypadków wykazały, że algorytm określania kandydujących wymagań czasowych znacząco redukuje liczbę wyrażeń czasowych, które muszą być przeanalizowane przez człowieka celem ostatecznego ustalenia listy wymagań czasowych wobec systemu. Jednocześnie algorytm analizy zależności czasowych w drzewach niezdatności MZPCALC_FULL ma złożoność czasową zbliżoną do rozmiaru produkowanego przez algorytm wyniku, co praktycznie wyklucza znaczący postęp w redukcji czasu przetwarzania bez ograniczenia rozmiaru rozwiązania. Wykazano, że algorytmy redukujące rozmiar rozwiązania charakteryzują się czasem przetwarzania niższym od powyższego algorytmu, co uzasadnia celowość ich zastosowania. Natomiast do celu redukcji nadmiarowych wyrażeń czasowych w wynikach analizy należy, jeśli to możliwe, używać algorytmu REDUCE_MZP, ze względu na krótszy czas obliczeń w najcięższych przypadkach. Algorytm REDUCE_MZP_ext należy stosować jedynie w przypadku MZP nie spełniających klasycznej definicji minimalności.

10 Podsumowanie

10.1 Osiągnięte wyniki

W wyniku przeprowadzonych badań osiągnięto następujące wyniki.

- Głównym wynikiem jest metoda TREM (Timing Requirements sElection Method) służąca identyfikacji wymagań czasowych dotyczących zdarzeń generowanych przez system sterujący systemu związanego z bezpieczeństwem na podstawie analizy drzew niezdatności dotyczących tego systemu. W ramach tej propozycji wprowadzono:
  1. Klasyfikację zdarzeń w systemach związanych z bezpieczeństwem umożliwiającą redukcję zarówno pracy analitycznej jak i nakładów obliczeniowych przy analizie drzew niezdatności wzbogaconych o czas.
  2. Sposób specyfikacji ograniczeń dziedzinowych umożliwiający weryfikację spełnialności zależności czasowych zarówno dla Minimalnych Zbiorów Przyczyn, jak i zidentyfikowanych wymaganiach czasowych.
  3. Cztery algorytmy wyznaczania Minimalnych Zbiorów Przyczyn rozszerzonych o czas.
  4. Dwa algorytmy redukcji nadmiarowości w wynikach przeprowadzanej analizy zależności czasowych.
  5. Algorytm określania kandydujących wymagań czasowych wobec analizowanego systemu na podstawie Minimalnych Zbiorów Przyczyn z zależnościami czasowymi, a uwzględniający klasyfikację zdarzeń oraz ograniczenia dziedzinowe.
- Poza zaproponowaniem metody TREM opracowano sposób jej walidacji przy wykorzystaniu podejścia GQM.
- Stworzono narzędzie analityczne PolymorphFT2 wspierające proponowaną metodę, które zostało użyte w ramach badań walidacyjnych.
- Przeprowadzono również, w ramach dwóch studiów przypadków, badania walidacyjne w odniesieniu do czterech różnych systemów związanych z bezpieczeństwem (specyfikacje tych systemów były publikowane w literaturze związanej z tematyką bezpieczeństwa).

10.2 Ocena wyników i ich oryginalność

Celem badań było zaproponowanie nowej metody identyfikowania uwarunkowanych czasem wymagań względem systemu komputerowego na podstawie analizy bezpieczeństwa środowiska, z którym system ten współpracuje.

Metoda TREM spełnia ten cel poprzez wywodzenie kandydujących wymagań wobec systemu, które określają wymagania czasowe wobec wydarzeń kontrolowalnych. Wydarzeniami kontrolowalnymi są w szczególności sygnały sterujące wydawane przez system komputerowy do urządzeń sterowanych. Wymagania te są wywodzone na podstawie analizy drzew niezdatności, będących uznaną metodą analizy bezpieczeństwa, rozszerzonych o definicje zależności czasowych pomiędzy zdarzeniami w poszczególnych bramkach. Metoda TREM jest propozycją oryginalną, która wykorzystuje jako punkt wyjścia modele i algorytm przedstawione w pracach [BCG91, Gor94, GW95 i War96].

Dodatkowym celem było opracowanie narzędzi wspomagających zastosowanie proponowanej metody w odniesieniu do rzeczywistych przypadków komputerowych systemów związanych z bezpieczeństwem. Cel ten zrealizowano poprzez stworzenie narzędzia PolymorphFT2. Jest to

narzędzie umożliwiające tworzenie drzew niezdatności, opisywanie zależności czasowych występujących w drzewie oraz określanie ograniczeń dziedzinowych jak również klasyfikowanie zdarzeń prostych. Dla tak stworzonego dokumentu możliwe jest wykonanie analizy określającej zarówno relacje czasowe pomiędzy wydarzeniami w poszczególnych Minimalnych Zbiorach Przyczyn, jak i kandydujące wymagania bezpieczeństwa. Wybór docelowych wymagań bezpieczeństwa jest wspierany poprzez tworzenie priorytetowej listy (uszeregowanie względem liczby scenariuszy zagrożeń, którym zapobiegają), możliwość zapisania wymagań czasowych przyjętych do realizacji oraz obrazowanie na liście scenariuszy hazardu które z nich nie będą mogły wystąpić dla przyjętej w danym momencie listy wymagań czasowych. Narzędzie wspiera więc wszystkie etapy metody, a ponieważ zostało użyte w opisywanych studiach przypadku, można postulować, że nadaje się do analizy drzew o rozmiarach faktycznie występujących w rzeczywistych przypadkach. Narzędzie PolymorphFT2 jest narzędziem oryginalnym stworzonym na potrzeby badań opisanych w pracy.

Z celem pracy związane były dwie tezy. Pierwsza z nich stanowi:

> 1. ***Proponowana metoda umożliwia analizę związanych z czasem scenariuszy hazardów pod kątem identyfikacji uwarunkowanych czasem wymagań bezpieczeństwa wobec systemów komputerowych.***

Zaproponowany algorytm *REQCALC* umożliwia analizę scenariuszy hazardów (zależności czasowych powiązanych z konkretnym Minimalnym Zbiorem Przyczyn) i określenie wymagań czasowych wobec zdarzeń kontrolowalnych. Zdarzenia kontrolowalne mogą być różnej natury, ale w szczególności można przyjąć, że sygnały sterujące wysyłane przez wbudowany system komputerowy należą do tej kategorii.

Dla przykładu można podać, że dla poszczególnych systemów analizowanych w ramach pierwszego studium przypadku (klasyfikacja zdarzeń była tam powiązana ze strukturą systemu; patrz podrozdział 9.1) uzyskano m.in. następujące czasowe wymagania kandydujące :

- System palnika gazowego
  $\forall e \in E14 \cdot duration(e) \leq T_g$,
  gdzie $E14$ to zdarzenie odpowiadające brakowi aktywności zapalnika, a T_g to czas po którego upłynięciu niespalany gaz może osiągnąć stężenie wybuchowe.
- System przejazdu kolejowego
  $\forall e_4 \in E_4 \cdot \exists e_{15} \in E_{15} \cdot start(e_4) + T_{cmin} - T_d - T_{bmax} - T_{emax} \geq start(e_{15})$,
  gdzie E_4 to zdarzenie polegające na wykryciu zbliżania się pociągu, a E_{15} to zdarzenie oznaczające wysłanie sygnału zamknięcia do zapór drogowych, przy założeniu, że samochody będą wjeżdżać na przejazd nawet po rozpoczęciu zamykania zapór. Stałe T_{cmin}, T_d, T_{bmax} oraz T_{emax} to stałe czasowe oznaczające odpowiednio: minimalny czas w czasie którego wykryty pociąg dotrze na przejazd, czas od odebrania sygnału do rozpoczęcia zamykania zapór, maksymalny czas zamykania zapór, maksymalny czas potrzebny aby samochód przejechał przez przejazd (zakładając, że nie ulegnie awarii).
- System monitoringu ruchu
  $\forall e_2 \in E_2 \cdot e_{15} \in E_{15} \cdot start(e_{15}) + T_{tmax} + T_d + T_{dl} \leq start(e2)$,
  gdzie E_2 to zdarzenie odpowiadające wystąpieniu niebezpiecznego zdarzenia na drodze (jest

ono klasyfikowane jako przewidywalne), E_{15} to zdarzenie polegające uruchomieniu procedury wysyłania ostrzeżeń do uczestników ruchu. Stałe czasowe T_{tmax}, T_d, T_{dl} oznaczają odpowiednio: maksymalny czas pomiędzy rozpoczęciem procedury wysyłania ostrzeżeń a odebraniem ostrzeżenia przez komputer pokładowy samochodu, czas od odebrania ostrzeżenia przez komputer pokładowy do wyświetlenia komunikatu kierowcy, czas przez który ostrzeżenie będzie wyświetlane.

Drugą tezę określono jako:

> 2. ***Proponowana metoda może być efektywnie zastosowana w odniesieniu do rzeczywistych przypadków komputerowych systemów związanych z bezpieczeństwem.***

Zaproponowany algorytm MZPCALC_FULL, służący do obliczania zależności czasowych w drzewach niezdatności, wywodzi się z algorytmu TGRAF-MZP (umożliwia jednak prowadzenie obliczeń dla alternatywnych zależności czasowych). Przeprowadzony eksperyment myślowy (podrozdział 5.2) pozwolił na znalezienie bardziej optymalnej strategii budowania rozwiązania niż w algorytmie wyjściowym (ten stosował strategię podstawową).

Ponadto przeprowadzone studium przypadku (patrz rozdział 9.1), umożliwia określenie, że wzrost czasu przetwarzania zależności czasowych jest proporcjonalny do otrzymanego wyniku (rośnie szybciej od liczby Minimalnych Zbiorów Przyczyn, ale wolniej niż łączna liczba wyrażeń czasowych we wszystkich Minimalnych Zbiorach Przyczyn), a pozostałe trzy algorytmy umożliwiają dalsze ograniczenie tego czasu. Te wszystkie fakty pozwalają postulować, że obliczanie zależności czasowych w metodzie TREM jest przeprowadzane efektywnie.

Drugim ważnym i złożonym obliczeniowo etapem metody jest obliczanie kandydujących wymagań czasowych dla uzyskanych scenariuszy hazardu. W drugim ze studiów przypadku (patrz rozdział 9.2) określono, że wzrost czasu obliczeń jest gorszy niż dla obliczania zależności czasowych i plasuje się pomiędzy łączna liczbą wyrażeń czasowych we wszystkich Minimalnych Zbiorach Przyczyn, a kwadratem liczby Minimalnych Zbiorów Przyczyn. Jednak zastosowanie innych niż MZPCALC_FULL algorytmów wyliczania zależności czasowych umożliwia ograniczenie czasu przetwarzania (poprzez ograniczenie rozmiaru danych wejściowych).

Co ważniejsze jednak, w pierwszym z przeprowadzonych studiów przypadków (podrozdział 9.1) ustalono, że algorytm ten umożliwia ograniczenie liczby wyrażeń czasowych w proponowanych wymaganiach czasowych w stosunku do ich liczby w zależnościach czasowych dla MZP nawet o 97%. Jest to istotne, gdyż stanowi o ilości pracy, którą będzie musiała wykonać osoba przeprowadzająca analizę bezpieczeństwa.

Przytoczone powyżej fakty uzasadniają twierdzenie, że cel niniejszej pracy został osiągnięty, a tezy udowodnione.

10.3 Upubliczniony dorobek badań

Badania prowadzone w ramach tej pracy zostały opisane w następujących publikacjach:

1) Gołaszewski, G. *Redukcja czasu analizy MZP przez ograniczenie rozmiaru rozwiązania*, In: *Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej*, 2013, Nr 36, pp. 61-64, ISSN 1425-5766 – w pracy opisano eksperyment mający na celu ustalenie czy i w jakiej mierze wprowadzenie w algorytmie wyznaczania Minimalnych Zbiorów Przyczyn z zależnościami czasowymi selektywnego obliczania wyników spowoduje przyspieszenie wykonywania obliczeń (podrozdział 9.2).
2) Gołaszewski, G. *Czasowe wymagania bezpieczeństwa wobec systemu monitoringu ruchu drogowego* ,In: V Krajowa Konferencja Technologie Informacyjne, Gdańsk 2007, Zeszyty Naukowe Wydz. ETI PG, Technologie Informacyjne, 2007, Tom 13, pp. 265-272. – praca wprowadza system monitoringu ruchu drogowego (podrozdział 9.1.2.3) i opisuje jak metoda TREM może być użyta do wywiedzenia wymagań czasowych dla tego systemu
3) Gołaszewski, G. *Dobór algorytmu do przetwarzania zależności czasowych w drzewach błędów*, In: *XIV Konferencja Systemy Czasu Rzeczywistego, Karpacz 2007, Systemy czasu rzeczywistego - Metody i zastosowania* , Polskie Towarzystwo Informatyczne, 2007, pp. 75-84. – praca opisuje eksperyment myślowy przeprowadzony w celu określenia najlepszej strategii dla wyliczania zależności czasowych dla wszystkich Minimlanych Zbiorów Przyczyn (podrozdział 5.2).
4) Gołaszewski, G. *Automatyzacja określania wymagań bezpieczeństwa na podstawie wyników analizy czasowej drzew błędów*, In: *Technologie Informacyjne, Maj 2006, Gdańsk, Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej*, 2006, vol. 11, pp. 631-638, ISBN 83-917681-8-X – praca wprowadza rozpatrywanie zależności czasowych jako predykatów logicznych, opisany został w niej sposób wywodzenia wymagań czasowych na podstawie wyników analizy drzewa niezdatności oraz klasyfikacji zdarzeń (podrozdział 7.1), przykłady opierały się o system palnika gazowego (podrozdział 9.1.2.1).
5) Gołaszewski, G.; Górski, J. *Hazard prevention by forced time constraints*, In: *IEEE Computer Society "Conference on Dependability of Computer Systems DepCoS – RELCOMEX'06"*, May 2006 Szklarska Poręba, pp. 84- 91, IEEE 2006, ISBN 978-0-7695-2565-5 doi: 10.1109/DEPCOS-RELCOMEX.2006.29 – w pracy wprowadzono podział zdarzeń na kategorie (podrozdział 4.3.1), opisano ograniczenia dziedzinowe oraz zaproponowano wyznaczanie wymagań czasowych w oparciu o analizę Minimalmych Zbiorów Przyczyn rozszerzonych o czas oraz postulowanej klasyfikacji (podrozdział 4.1), przykłady oparto o system przejazdu kolejowego (podrozdział 9.1.2.2).
6) Gołaszewski, G. *Tool support for ECSDM fault tree methodology* , In: *Proceedings of TEHOSS 2005 : IEEE International Conference on Technologies for Homeland Security and Safety,* Gdańsk, Poland, September 2005, pp. 243-248. – w pracy przedstawiono notację ECSDM, zarówno model dynamiczny(podrozdział 3.2), jak i statyczny (podrozdział 3.3), jednak główny nacisk położono na wczesną wersję narzędzia analitycznego (ostateczna wersja opisana w podrozdziale 8.2), którego funkcjonalność omawiano w oparciu o system palnika gazowego (podrozdział 9.1.2.1).
7) Gołaszewski, Grzegorz *Dowody lematów o które oparto metodę TREM* Raport techniczny nr. 3/2017 wydziału ETI PG, Gdańsk 2017 – w pracy tej zawarte zostały dowody dwóch twierdzeń, na które powołano się w tej rozprawie

8) Gołaszewski, Grzegorz *Słownik normalizacji wyrażeń ECSDM* Raport techniczny nr. 4/2017 wydziału ETI PG, Gdańsk 2017 – raport zawiera pełny słownik wyrażeń ECSDM wraz z odpowiadającymi im wyrażeniami w postaci normalnej
9) Gołaszewski, Grzegorz *Formalna definicja algorytmów wykorzystywanych w metodzie TREM* Raport techniczny nr. 5/2017 wydziału ETI PG, Gdańsk 2017 – raport zawiera specyfikację w pseudokodzie algorytmów wykorzystywanych w metodzie TREM

10.4 Dalsze prace

W zakresie dalszych prac mieści się wsparcie dla specyfikacji i analizy wielu wystąpień (akcji) pojedynczego zdarzenia. W obecnym momencie nie jest to wspierane i aby wyspecyfikować dwa wystąpienia danego zdarzenia trzeba je dwukrotnie modelować w drzewie. Postulowane rozszerzenie umożliwiłoby specyfikowanie dla wybranych bramek drzewa niezdatności zależności czasowych obejmujących dwa wystąpienia danego zdarzenia (np. dla systemu palnika gazowego {podrozdział 9.1.2.1} dałoby się wyspecyfikować, że niebezpieczne stężenie gazy wystąpi nie tylko gdy będzie dopływ gazy bez zapalonego płomienia przez więcej niż *Tg*, ale również jeśli będzie on krótszy niż Tg, ale znowu wystąpi po czasie krótszym niż *Tw*{czas całkowitego wywietrzenia pieca}) bez kopiowania poddrzewa opisującego to zdarzenie. Możliwe byłoby również uniknięcie kopiowania poddrzewa w innej bramce, jeśli rozpatrywane zdarzenie przyczynia się do wystąpienia więcej niż jednego zdarzenia wyższego poziomu. Modyfikacja taka nie wymagałaby większych zmian w zaproponowanych algorytmach (trzeba zadbać, aby dwa wystąpienia tego samego zdarzenia były tożsame albo rozłączne). Związana byłaby głównie z modyfikacją narzędzia wspierającego metodę. Wprowadzona zmiana umożliwiłaby również wprowadzenie do metody TREM komponentów (patrz [KLM03]).

Warte rozważenia jest również rozszerzenie zaproponowanej klasyfikacji zdarzeń. Możliwe byłoby na przykład wprowadzenie rozróżnienia zdarzeń kontrolowalnych, które mogą być cyklicznie wywoływane bez szkody dla misji systemu i powodowania zagrożeń. To pozwoliłoby na stworzenie adaptowalnego algorytmu obliczania wybranych zależności czasowych. Dla zdarzeń kontrolowalnych, których nie można wyzwalać cyklicznie, algorytm taki odrzucałby MZP zawierające takie zdarzenie oraz same zdarzenia niekontrolowane (jak MZPCALC_PAIR), a dla zdarzeń kontrolowalnych odpalanych cyklicznie by je pozostawiał (jak MZPCALC_CONTROLABLE).

Bibliografia

[ALRL04] Avizienis, Algirdas; Laprie, Jean-Claude; Randell, Brian and Landwehr, Carl *Basic concepts and taxonomy of dependable and secure computing*. *IEEE Transactions on Dependable and Secure Computing*, 2004, vol. 1, no. 1,. pp. 11-33, doi: 10.1109/TDSC.2004.2

[Amb04] Ambler, Scott W. *The object primer: Agile model-driven development with UML 2.0*. Cambridge University Press, 2004. ISBN: 0-521-54018-6

[AMC87] Ajmone Marsan, M.; Chiola, G. *On Petri nets with deterministic and exponentially distributed firing times*. In Rozenberg, Grzegorz *Advances in Petri Nets 1987*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, Lecture Notes in Computer Science vol. 266, pp 132-145, ISBN 978-3-540-47743-3

[AP95] Aizenstein, Howard; Pitt, Leonard *On the Learnability of Disjunctive Normal Form Formulas*. *Machine Learning*, 1995, vol. 19, iss. 3, pp 183–208, doi:10.1023/A:1022697326858

[AR02] Andrews, John D.; & Ridley, Louise M. *Application of the cause-consequence diagram method to static systems*. *Reliability Engineering and System Safety*, January 2002, vol. 75, iss. 1, pp. 47-58., ISSN 0951-8320

[ATSB08] Australian Transport Safety Bureu *Qantas Airbus A330 accident Media Conference*, Media relase 14 Październik 2008[dostęp: 20 VI 2016, http://www.atsb.gov.au/newsroom/2008/release/2008_43.aspx]

[AZ11] Aliee , Hananeh; Zarandi , Hamid R. *Fault tree analysis using stochastic logic: A reliable and high speed computing*. In: *Reliability and Maintainability Symposium (RAMS)*, 2011 Proceedings - Annual, Lake Buena Vista, FL, IEEE 2011, pp. 1-6. doi: 10.1109/RAMS.2011.5754466

[BCG91] Bloomfield, R. E.; Cheng, J. H.; Górski J. *Towards A Common Safety Description Model*. In: Lindeberg, Johan F. *Proceedings of Safecomp'91*, Oxford, Pergamon Press, 1991

[BFGP99] Bobbio, A.; Franceschinis, G.; Gaeta, R. and Portinale, L. *Exploiting Petri nets to support fault tree based dependability analysis*. In: *Petri Nets and Performance Models, 1999. Proceedings. The 8th International Workshop on*, Zaragoza, IEEE, September 1999, pp. 146-155. doi: 10.1109/PNPM.1999.796561

[BLM09] Babczynski, Tomasz; Lukowicz, Miroslaw and Magott, Jan *Selection of Tripping Times for Distance Protection Using Probabilistic Fault Trees with Time Dependencies*. In: *Dependability of Computer Systems, 2009. DepCos-RELCOMEX '09. Fourth International Conference on*, Brunow, IEEE, June 2009, pp. 315-323, doi: 10.1109/DepCoS-RELCOMEX.2009.18

[BLM10] Babczynski, Tomasz; Lukowicz, Miroslaw and Magott, Jan *Time Coordination of Distance Protections Using Probabilistic Fault Trees With Time Dependencies*. *IEEE Transactions on Power Delivery*, July 2010, vol. 25, no. 3, pp. 1402-1409, doi: 10.1109/TPWRD.2010.2046342

[BR04] Bobbio, A. and Raiteri, D. C. *Parametric fault trees with dynamic gates and repair boxes*. In: *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, January 2004, pp. 459-465, doi: 10.1109/RAMS.2004.1285491

[Bri05] Briggs, Helen *Cryosat rocket fault laid bare*. BBC News, 27 October 2005 [dostęp: 20 VI 2016, http://news.bbc.co.uk/2/hi/science/nature/4381840.stm]

[Bur95] Burke, Dennis *All Circuits are Busy Now: The 1990 AT&T Long Distance Network Collapse*. California Polytechnic State University, November 1995 [dostęp: 28 VI 2016, http://users.csc.calpoly.edu/~jdalbey/SWE/Papers/att_collapse.html]

[Car69] Carre, Bernard A. *A matrix factorization method for finding optimal paths through networks*. In: *IEE Conf. Publ.(Comput.-Aided Design)*. 1969, vol. 51., pp. 388-397

[Car71] Carré, Bernard A. *An algebra for network routing problems*. In: *IMA Journal of Applied Mathematics*, 1971, vol 7, no.3, pp. 273-294

[CHR91] Chaochen, Zhou; Hoare, Charles Anthony Richard; Ravn, Anders P. *A calculus of durations*. In: *Information processing letters*, 1991, vol. 40 iss. 5, pp. 269-276, ISSN 0020-0190, doi: 10.1016/0020-0190(91)90122-X.

[CLRS05] Cormen, T.H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. *Wprowadzenie do algorytmów*, Wydawnictwa Naukowo-Techniczne, Warszawa 2005.

[CM02] Čepin, Marko; Mavko, Borut. *A dynamic fault tree*. In: *Reliability Engineering & System Safety*, January 2002, vol. 75, iss. 1, pp. 83-91, ISSN 0951-8320, doi: 10.1016/S0951-8320(01)00121-1

[Dal99] Dalcher, Darren. *Disaster in London. The LAS case study*. In: *Engineering of Computer-Based Systems, 1999. Proceedings. ECBS'99. IEEE Conference and Workshop on*. IEEE, 1999, pp. 41-52 doi:

10.1109/ECBS.1999.755860

[DBB92] Dugan, Joanne Bechta; Bavuso, Salvatore J.; Boyd, Mark A. *Dynamic fault-tree models for fault-tolerant computer systems*. In: *IEEE Transactions on reliability*, Sep 1992, vol. 41, no. 3, pp. 363-377,. doi: 10.1109/24.159800.

[DGS09] Rao, K. Durga; Gopika, V.; Sanyasi Rao, V.V.S.; Kushwaha, H.S.; Verma, A.K.; Srividya, A. *Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment*. In: *Reliability Engineering & System Safety*, April 2009, vol. 94, iss. 4, pp. 872-883, ISSN 0951-8320, doi: 10.1016/j.ress.2008.09.007.

[Dij59] Dijkstra, Edsger W. *A note on two problems in connexion with graphs*. In: *Numerische mathematik*, Dec. 1959, vol. 1, iss. 1, pp 269-271,

[EC96] Report by the Inquiry Board, ESA and CNES, *ARIANE 5. Flight 501 Failure*, Paris 19 July 1996, [dostęp: 25 IV 2017, https://www.ima.umn.edu/~arnold/disasters/ariane5rep.html]

[Eri99] Ericson II, C. A. *Fault Tree Analysis-A History*, In: *17th International System Safety Conference, System Safety Society*, 1999, [dostęp:25 IV 2017, http://www.relken.com/sites/default/files/Seminal%20Documents/ericson-fta-history.pdf]

[Fak14] Urządzenie radio-stop zablokowało pociągi i nie doszło do wypadku, Fakt.pl 22 IX 2014 [dostęp:25 IV 2017, http://www.fakt.pl/wydarzenia/polska/urzadzenie-radio-stop-zablokowalo-pociagi-i-nie-doszlo-do-wypadku/xd7dhmg]

[FD96] Finkelstein, Anthony; Dowell, John. *A comedy of errors: the London Ambulance Service case study*. In: *Proceedings of the 8th International Workshop on Software Specification and Design*. IEEE Computer Society, 1996. p. 2.

[Flo62] Floyd, Robert W., *Algorithm 97: shortest path.* In: *Communications of the ACM*, 1962, vol. 5, no. 6, p. 345

[GD97] Gulati, Rohit; Dugan, Joanne Bechta. *A modular approach for analyzing static and dynamic fault trees*. In: *Reliability and Maintainability Symposium. 1997 Proceedings, Annual.* IEEE, 1997, p. 57-63.

[GG06] Gołaszewski, G.; Górski, J. *Hazard prevention by forced time constraints*, In: *Proceedings of International Conference on Dependability of Computer Systems DepCoS – RELCOMEX'06*, Szklarska Poręba 25-27 May, 2006, IEEE Computer Society, pp. 84-91.

[GGA95] Garrett, Christopher James; Guarro, Sergio B.; Apostolakis, George E. *The dynamic flowgraph methodology for assessing the dependability of embedded software systems*. In: *IEEE Transactions On Systems, Man, and Cybernetics*, May 1995, vol. 25, no. 5, pp. 824-840. doi: 10.1109/21.376495

[GMA95] Górski, J.; Magott, J.; Wardziński, A. *Modeling Fault Trees Using Petri Nets*, In: Rabe, G. *Proceedings of the 14th International Conference on Computer Safety, Reliability and Security SAFECOMP'95*, Springer London, 1995, pp. 90-100.

[Gol05] Gołaszewski, G. *Tool support for ECSDM fault tree methodology*, In: Stepnowski, A.; Ruciński, A.; Kosmowski, K, *Proceedings of TEHOSS 2005 : IEEE International Conference on Technologies for Homeland Security and Safety*, Gdańsk, Poland, September 28-30, 2005, Gdańsk University of Technology 2005, pp. 243-248

[Gol06] Gołaszewski, G. *Automatyzacja określania wymagań bezpieczeństwa na podstawie wyników analizy czasowej drzew błędów*, In: *Zeszyty Naukowe Wydziału Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej*, vol. 11 (2006), pp. 631-638, ISBN 83-917681-8-X

[Gol07] Gołaszewski, G. *Czasowe wymagania bezpieczeństwa wobec systemu monitoringu ruchu drogowego*, In: *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*,vol. 13 (2007), pp. 265-272, ISBN 978-83-60779-01-9

[Gol07b] Gołaszewski, G. *Dobór algorytmu do przetwarzania zależności czasowych w drzewach błędów* In: Huzar, Z.; Mazur, Z. *Systemy czasu rzeczywistego : Metody i zastosowania : praca zbiorowa*, Polskie Towarzystwo Informatyczne., Warszawa : WKŁ, 2007, pp. 75-84, ISBN 978-83-206-1658-3

[Gol13] Gołaszewski, G. *Redukcja czasu analizy MZP przez ograniczenie rozmiaru rozwiązania*. In: *Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej*, 2013, pp. 61 – 64

[Gol17a] Gołaszewski, Grzegorz *Dowody lematów o które oparto metodę TREM* Raport techniczny nr. 3/2017 Wydziału ETI PG, Gdańsk 2017

[Gol17b] Gołaszewski, Grzegorz *Słownik normalizacji wyrażeń ECSDM* Raport techniczny nr. 4/2017 Wydziału ETI PG, Gdańsk 2017

[Gol17c] Gołaszewski, Grzegorz *Formalna definicja algorytmów wykorzystywanych w metodzie TREM* Raport

techniczny nr. 5/2017 Wydziału ETI PG, Gdańsk 2017

[Gor94] Górski, Janusz *Extending safety analysis techniques with formal semantics.* In: *Technology and Assessment of Safety-Critical Systems*. Springer London, 1994. pp. 147-163

[GW95] Górski, Janusz; Wardziński, Andrzej *Formalising fault trees.* In: *Achievement and Assurance of Safety.* Springer London, 1995. p. 311-327

[GW97] Górski, Janusz; Wardziński, Andrzej *Timing aspects of fault tree analysis of safety critical systems.* In: *Safer Systems, Proceedings of the Fifth Safety-critical Systems Symposium*, Brighton 4-6 February 1997, London Springer, 1997, pp. 231-244, isbn 978-1-4471-0975-4, doi 10.1007/978-1-4471-0975-4_14

[HA88] Hura, G. S.; Atwood, J. W. *The use of Petri nets to analyze coherent fault trees* In: *IEEE Transactions on Reliability*, Dec 1988, vol. 37, no. 5, pp. 469-474, doi: 10.1109/24.9864

[Hag06] Hagerup, T. *Simpler Computation of Single-Source Shortest Paths in Linear Average Time* In: *Theory of Computing Systems*, Jan/Feb2006, vol. 39 iss. 1, pp. 113-120; doi: 10.1007/s00224-005-1260-0

[HC07] Huang, Chin-Yu; Chang, Yung-Ruei *An improved decomposition scheme for assessing the reliability of embedded systems by using dynamic fault trees* In: *Reliability Engineering & System Safety*, 2007, vol. 92, iss. 10, pp. 1403-1412, ISSN 0951-8320, doi: 10.1016/j.ress.2006.09.008.

[HMSO92] *The tolerability of risk from nuclear power stations*, HMSO, London, 1992, ISBN 0118863681

[HRS98] Hansen, Kirsten M.; Ravn, Anders P.; Stavridou, Victoria *From safety analysis to software requirements.* In: *IEEE Transactions on Software Engineering*, July 1998, vol. 24, no. 7, pp. 573-584, doi: 10.1109/32.708570

[HSC98] *The use of computers in safety-critical applications*, Health and Safety Comission, London, 1998 (ISBN 0 7176 1620 7)

[HSE01] *Reducing risks, protecting people. HSE's decision-making process*, HSE, London 2001 (ISBN 0 7176 2151 0)

[IAEA] *Lista aktualnych standardów bezpieczeństwa*, Międzynarodowa Agencja Energii Atomowej, [dostęp:25 IV 2017, http://www-ns.iaea.org/standards/documents/pubdoc-list.asp?s=11&l=83]

[IAEA15] *The Fukushima Daiichi Accident, Report by the director general*, IAEA, 2015 [dostęp:25 IV 2017, http://www-pub.iaea.org/books/IAEABooks/10962/The-Fukushima-Daiichi-Accident]

[IR5] *Instrukcja o użytkowaniu urządzeń radiołączności pociągowej Ir-5 (R-12)*, PKP PLK, Warszawa 2014

[ISO8124] ISO 8124 Safety of toys, Części 1-8 (lata publikacji 2010-2016)

[Jar96] Jarmuż, Piotr *Analiza bezpieczeństwa systemu czasu rzeczywistego, Praca magisterska*, Wydział Informatyki EFP, 1996

[Joh77] Johnson, Donald B. *Efficient algorithms for shortest paths in sparse networks* In: *Journal of the ACM (JACM)*, 1977, vol. 24, no. 1, pp. 1-13

[Jon90] Jones, C.B. *Systematic Software Development Using VDM (2nd Edition)*, Prentice Hall International, 1990

[KGF07] Kaiser, B.; Gramlich, C; Förster, M. *State/event fault trees—A safety analysis model for software-controlled systems*, In: *Reliability Engineering & System Safety*, November 2007, vol. 92, iss. 11, pp. 1521-1537, ISSN 0951-8320, doi:10.1016/j.ress.2006.10.010.

[KLM03] Kaiser, Bernhard; Liggesmeyer, Peter; Mäckel, Oliver *A new component concept for fault trees*. In: *Proceedings of the 8th Australian workshop on Safety critical systems and softwa*re - Volume 33 (SCS '03), Australian Computer Society, Inc., 2003, vol. 33, pp. 37-46

[Kor78] Korzan, Bohdan *Elementy teorii grafów i sieci: metody i zastosowania*, Wydawnictwa Naukowo-Techniczne, Warszawa 1978

[KSW14] Klüppelberg, C.; Straub, D.; Welpe, I. M. *Risk - A Multidisciplinary Introduction*, Springer 2014, ISBN 978-3-319-04486-6

[LC09] Li, Qing; Chen, Yu-Liu, *Data flow diagram*. In: *Modeling and Analysis of Enterprise and Information Systems*, Springer Berlin Heidelberg 2009, pp. 85-97, DOI 10.1007/978-3-540-89556-5_4, ISBN 978-3-540-89555-8

[LMS11] Lukowicz, Miroslaw; Magott, Jan; Skrobanek, Pawel *Selection of minimal tripping times for distance protection using fault trees with time dependencies*, In: *Electric Power Systems Research*, July 2011, vol. 81, iss. 7, pp. 1556-1571, ISSN 0378-7796, doi: 10.1016/j.epsr.2011.03.003.

[LT93] Leveson, N. G.; Turner, C. S. *An investigation of the Therac-25 accidents* In: *Computer*, July 1993, vol. 26, no. 7, pp. 18-41, doi: 10.1109/MC.1993.274940

[MDA02] L. Meshkat, J. B. Dugan and J. D. Andrews, "Dependability analysis of systems with on-demand and active failure modes, using dynamic fault trees," in IEEE Transactions on Reliability, vol. 51, no. 2, pp. 240-251, Jun 2002. doi: 10.1109/TR.2002.1011531

[MDCS98] Manian, R.; Dugan, J. Bechta; Coppit, D.; Sullivan, K. J. *Combining various solution techniques for dynamic fault tree analysis of computer systems* In: *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*, Washington, DC, 1998, pp. 21-28. doi: 10.1109/HASE.1998.731591

[MIL29] *Procedures for Performing a Failure Mode, Effects and Criticaility Analysis.* U.S. Department of Defense. Washington, DC 20301. 1980. MIL–STD–1629A. [dostęp: 25 IV 2017, https://src.alionscience.com/pdf/MIL-STD-1629RevA.pdf]

[MLST12] Magott, J.; Lewiński, A.; Skrobanek, P.; Toruń, A. *The FTTD method application to the safety analysis of Changeable Block Distance System* In: *International Conference on Transport Systems Telematics*, Springer Berlin Heidelberg, October 2012, pp. 267-275

[MNOS07] Merz, S.; Nipkow, T.; Ortmeier, Frank; Schellhorn, Gerhard *Proceedings of the 6th International Workshop on Automated Verification of Critical Systems (AVoCS 2006) Formal Fault Tree Analysis - Practical Experiences, Electronic Notes* In: *Theoretical Computer Science*, 2007, Volume 185, Pages 139-151, ISSN 1571-0661, doi: 10.1016/j.entcs.2007.05.034

[MNSW08] Magott, J.; Nowakowski, T.; Skrobanek, P.; Werbińska, S. *Analysis of possibilities of timing dependencies modelling – example of logistic support system* In: *European Safety and Reliability Association Conferen*ce, *ESREL, 2008*, Valencia, Spain, Leiden: Taylor and Francis, 2008, pp. 1055-10

[MS00] Magott, J.; Skrobanek, P. *A Method of Analysis of Fault Tree with Time Dependencies* In: *Proc. SAFECOMP 2000, Rotterdam, The Netherlands, LNCS*, Springer-Verlag, 2000, vol. 1943, pp. 176-186.

[MS02] Magott, J.; Skrobanek, P. *Method of time Petri net analysis for analysis of fault trees with time dependencies* In: *IEE Proceedings - Computers and Digital Techniques*, Nov 2002, vol. 149, no. 6, pp. 257-271, doi: 10.1049/ip-cdt:20020804

[MT95] Malhotra, M.; Trivedi, K. S. *Dependability modeling using Petri-nets* In: *IEEE Transactions on Reliability*, Sep 1995, vol. 44, no. 3, pp. 428-440, doi: 10.1109/24.406578

[MTJ14] Mouaffo, A.; Taibi, D.; Jamboti, K. *Controlled experiments comparing fault-tree-based safety analysis techniques* In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*, ACM 2014, New York, NY, USA, art. 46, 10 pages, doi: 10.1145/2601248.2601255

[MXDO03] Meshkat, L.; Xing, L.; Donohue, S.K.; Ou, Y. *An overview of the phase-modular fault tree approach to phased mission system analysis* In: *Proceedings of the first international conference on space mission challenges for information technology (SMC-IT)*, July 2003, Pasadena, CA: JPL Publication 03-13A, Jet Propulsion Laboratory, California Institute of Technology; pp. 393–8.

[NASA02] Stamatelatos, M.; Vesely, W. *Fault Tree Handbook with Aerospace Applications*, NASA Office of Safety and Mission Assurance, August, 2002

[NCAP16] Latest Safety Ratings, Euro NCAP 2016 [dostęp: 25 IV 2017, http://www.euroncap.com/en/ratings-rewards]

[Neu86] Neumann, P. G. *Some Computer-Related Disasters and Other Egregious Horrors* In: *IEEE Aerospace and Electronic Systems Magazine*, Oct. 1986, vol. 1, no. 10, pp. 18-19, doi: 10.1109/MAES.1986.5004967

[Nie71] Nielsen, D.S. *The cause/consequence diagram method as a basis for quantitative accident analysis (Riso-M; No. 1374)*, Danish Atomic Energy Commission, 1971

[NUREG0492] Vesely, W. E.; Goldberg, F. F.; Roberts, N. H.; Haasl, D. F. *Fault Tree Handbook*, U.S. Nuclear Regulatory Commission, January 1981

[Pal02] Palshikar, Girish Keshav *Temporal fault trees* In: *Information and Software Technology*, 15 March 2002, vol. 44, iss. 3, pp. 137-150, ISSN 0950-5849, doi: 10.1016/S0950-5849(01)00223-3.

[PAM91] Parnas, D. L.; Asmis, G. J. K.; Madey, J. *Assessment of safety-critical software in nuclear power plants* In: *Nuclear safety*, 1991, vol. 32, iss. 2, pp. 189-198.

[PD-AP-1312] *NASA Preferred Reliability Practices, The team approach to fault-tree analysis*, Practice no. PD-AP-1312, NASA

[PN50128] PN-EN 50128:2011 Zastosowania kolejowe -- Systemy łączności, przetwarzania danych i sterowania ruchem -- Oprogramowanie kolejowych systemów sterowania i zabezpieczenia

[PN60812] PN-EN 60812:2009 Techniki analizy nieuszkadzalności systemów. Procedura analizy rodzajów i skutków uszkodzeń (FMEA)

[PN61508] PN-EN 61508:2010 Bezpieczeństwo funkcjonalne elektrycznych/elektronicznych/programowalnych elektronicznych systemów związanych z bezpieczeństwem

[PN61511] PN-EN 61511 Bezpieczeństwo funkcjonalne; Przyrządowe systemy bezpieczeństwa dla sektora przemysłu

procesowego
[PN61800] PN-EN 61800-5-2:2007 Elektryczne układy napędowe mocy o regulowanej prędkości -- Część 5-2: Wymagania dotyczące bezpieczeństwa -- Funkcjonalne
[PN62061] PN-EN 62061:2008 Bezpieczeństwo maszyn -- Bezpieczeństwo funkcjonalne elektrycznych, elektronicznych i elektronicznych programowalnych systemów sterowania związanych z bezpieczeństwem
[PN71] PN-EN 71 Bezpieczeństwo zabawek, Części 1-13 (lata publikacji 2005-20014)
[PR05] Pettie, S.; Ramachandran, V. *A Shortest Path Algorithm for Real-Weighted Undirected Graphs* In: *SIAM Journal on Computing*, Society for Industrial and Applied Mathematics 2005, vol. 34, no. 6, pp. 1398–1431.
[RLT78] Randell, B.; Lee, P.; Treleaven, P. C. *Reliability Issues in Computing System Design* In: *ACM Comput. Surv.*, June 1978, vol. 10, iss.2, pp. 123-165
[SB99] Van Solingen, Rini; Berghout, Egon *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*, McGraw-Hill, 1999
[Skr05] Skrobanek, P *Analiza zależności czasowych w drzewach niezdatności (rozprawa doktorska)*, Instytut Informatyki, Automatyki i Robotyki Politechniki Wrocławskiej, Wrocław, czerwiec 2005
[SMC74] Stevens, W. P.; Myers, G. J.; Constantine, L.L. *Structured design*, In: *IBM Systems Journal*, 1974, vol.13, no.2, pp.115-139, ISSN: 0018-8670
[STR02] Schellhorn, G.; Thums, A.: Reif, W. *Formal Fault Tree Semantics*. In: *Proceedings of the 6th world conference on integrated design & process technology*, Pasadena, CA, July 2002, pp. 1-8.
[TD04] Tang, Zhihua; Dugan, J. B. *Minimal cut set/sequence generation for dynamic fault trees*, In: *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, 2004, pp. 207-213. doi: 10.1109/RAMS.2004.1285449
[Tre16] *Introduction to Decision Trees*. TreePlan Software [dostęp: 25 IV 2017, http://treeplan.com/chapters/introduction-to-decision-trees.pdf]
[Tvn11] *"Radio stop" dla pociągów. Była zabawa, będzie proces*, TVN24.pl 22 IV 2011 [dostęp: 25 IV 2017, http://www.tvn24.pl/wiadomosci-z-kraju,3/radio-stop-dla-pociagow-byla-zabawa-bedzie-proces,168747.html]
[Vas97] Gerogiannis, V. C.; Caragiannis, I. E.; Tsoukarellas, M. A. *A general framework for applying safety analysis to safety critical real-time applications using fault trees*, In: *Real-Time Systems, 1997. Proceedings., Ninth Euromicro Workshop on*, Toledo, 1997, pp. 168-175. doi: 10.1109/EMWRTS.1997.613778
[War62] Warshall, Stephen *A theorem on boolean matrices*, In: *Journal of the ACM*, 1962, vol. 9 no. 1, pp. 11-12
[War96] Wardziński, Andrzej *Analiza drzew błędów systemów komputerowych związanych z bezpieczeństwem*, Dissertation Thesis, Technical University of Gdańsk, June 1996
[WJS05] Wang, I.; Johnson, E. L.; Sokol, J. S. *Multiple Pairs Shortest Path Algorithm*, In: *Transportation Science*; Nov 2005, vol. 39, iss. 4, ABI/INFORM Global pg. 465.
[WM00] Wijayarathna , P. G.; Maekawa, M. *Extending fault trees with an AND-THEN gate*, In: *Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on*, San Jose, CA, 2000, pp. 283-292. doi: 10.1109/ISSRE.2000.885879
[WP09] Walker, Martin; Papadopoulos, Yiannis *Qualitative temporal analysis: Towards a full implementation of the Fault Tree Handbook*, In: *Control Engineering Practice*, October 2009, vol. 17, iss. 10, pp. 1115-1125, ISSN 0967-0661, doi: 10.1016/j.conengprac.2008.10.003.
[WP10] Walker, Martin; Papadopoulos, Yiannis *A hierarchical method for the reduction of temporal expressions in Pandora*. In: *Proceedings of the First Workshop on DYnamic Aspects in DEpendability Models for Fault-Tolerant Systems (DYADEM-FTS '10)*, ACM 2010, New York, NY, USA, 7-12. doi: 10.1145/1772630.1772634
[XHHWZ13] Xu, Bingfeng; Huang, Zhiqiu; Hu, Jun; Wei, Ou; Zhou, Yu *Minimal cut sequence generation for state/event fault trees*. In *Proceedings of the 2013 Middleware Doctoral Symposium (MDS '13)*, ACM 2013, New York, NY, USA, , art. 3 , 6 pages, doi: 10.1145/2541534.2541592
[XMD07] Xing, Liudong; Meshkat, Leila; Donohue, Susan K. *Reliability analysis of hierarchical computer-based systems subject to common-cause failures*, In: *Reliability Engineering & System Safety*, March 2007, vol. 92, iss. 3, pp. 351-359, ISSN 0951-8320, doi: 10.1016/j.ress.2006.04.010.
[XML08] *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation, November 2008, [dostęp: 25 IV 2017, http://www.w3.org/TR/2008/REC-xml-20081126/]
[YC79] Yourdon, E.; Constantine, L. L. *Structured design: Fundamentals of a discipline of computer program and systems design (Vol. 5)*, Prentice-Hall 1979, ISBN-10: 0138544719
[YGA95] Yau, M.; Guarro, S.; Apostolakis, G. *Space System Applications of Risk Assessment. Demonstration of the*

Dynamic Flowgraph Methodology using the Titan II Space Launch Vehicle Digital Flight Control System, In: *Reliability Engineering & System Safety*, 1995, vol. 49, iss. 3, pp. 335-353, ISSN 0951-8320, doi: 10.1016/0951-8320(95)00050-C

[YAG98] Yau, Michael; Apostolakis, George; Guarro, Sergio *The use of prime implicants in dependability analysis of software controlled systems*, In: *Reliability Engineering & System Safety*, October–November 1998, vol. 62, iss. 1–2, pp. 23-32, ISSN 0951-8320, doi: 10.1016/S0951-8320(98)00002-7

[YH06] Yang, T.-P.; Ho, L.-H. *The Comparison and Study of Shortest Path Algorithm in Heaps - Using a Taiwan Route Map as Example*, In: *Journal of American Academy of Business*, Cambridge; Mar 2006; vol. 9, iss.1, ABI/INFORM Global, pg. 139.

Dodatki

D.1. Słownik pojęć

aktywna niezdatność	niezdatność powodująca w danym momencie błąd w systemie
bezpieczeństwo	wolność od nieakceptowalnych ryzyk
błąd	część stanu systemu, która może doprowadzić do uszkodzenia
bramka	element drzewa niezdatności reprezentujący zależność przyczynową pomiędzy zdarzeniami wyjściowym (skutek) a wejściowymi (przyczyny). W zależności od typu bramki wszystkie (AND) bądź jedno (XOR) ze zdarzeń wejściowych musi wystąpić, aby wystąpiło zdarzenie wyjściowe
bramka AND	bramka drzewa niezdatności wskazująca, że zdarzenie wyjściowe zachodzi tylko gdy wszystkie zdarzenia wejściowe zajdą
bramka OR	bramka drzewa niezdatności wskazująca, że zdarzenie wyjściowe zachodzi gdy zajdzie dowolne ze zdarzeń wejściowych
bramka warunkowa	bramka drzewa niezdatności wskazująca, że zdarzenie wyjściowe zachodzi gdy zajdzie zdarzenie wejściowe (jest tylko jedno) oraz spełniony jest specjalny warunek. Warunek modelowany jest przez zdarzenie warunkujące narysowane na prawo od bramki.
drzewo niezdatności	struktura opisująca przyczyny wystąpienia konkretnego zdarzenia poprzez wielokrotne opisanie bezpośrednich przyczyn zdarzeń (przy użyciu bramek)
dające się przewidzieć sposoby niewłaściwego wykorzystania	(ang. reasonably foreseeable misuse) wykorzystanie produktu, procesu bądź usługi w sposób nie przewidziany przez dostawcę, ale wynikający z dającego sie przewidzieć zachowania ludzkiego
E/E/PE	(electric/electronic/programmable electronic) elektryczny/elektroniczny/programowalny
elementowa funkcja bezpieczeństwa	(ang. element safety function) - część funkcji bezpieczeństwa implementowana przez poszczególny element systemu
funkcja bezpieczeństwa	funkcja mająca być zaimplementowana przez system E/E/PE związany z bezpieczeństwem bądź inny sposób redukcji ryzyka, której celem jest osiągnięcie lub utrzymanie systemu w stanie bezpiecznym, specyfikowana w odniesieniu do konkretnego niebezpiecznego zdarzenia

funkcja bezpieczeństwa oprogramowania	(ang. software safety function) funkcja bezpieczeństwa realizowana przez oprogramowanie
hazard	patrz zagrożenie
kandydujące wymaganie czasowe	wymaganie czasowe wytypowane za pomocą algorytmu REQCALC jako wymagania potencjalnie zwiększające bezpieczeństwo systemu. Wykonalność takiego wymagania musi zostać zweryfikowana przez analityka
konieczne zmniejszenie ryzyka	(ang. necessary risk reduction) poziom zmniejszenia ryzyka konieczny do uzyskania ryzyka tolerowalnego
losowe uszkodzenie sprzętu	(ang. random hardware failure), występujące w dowolnym momencie czasu uszkodzenie powstałe na skutek jednego lub większej liczby procesów degradacji zachodzących w sprzęcie
Minimalny Zbiór Przyczyn (MZP)	taki zbiór zdarzeń prostych, które doprowadzą do wystąpienia zdarzenia szczytowego drzewa niezdatności i z którego nie można usunąć żadnego zdarzenia bez naruszania poprzedniej własności, w przypadku metody TREM zawiera również zależności czasowe między zdarzeniami, które muszą zostać spełnione, aby wystąpiło zdarzenie szczytowe
nadmiarowy T-graf	T-graf, który można usunąć ze zbioru T-grafów konkretnego MZP bez zmiany wartościowania sumy tych zbiorów
niebezpieczna sytuacja	okoliczności w których ludzie, mienie bądź środowisko wystawieni są na jedno bądź więcej zagrożeń
niebezpieczne zdarzenie	zdarzenie które może skutkować szkodą
nienaruszalność bezpieczeństwa	(ang. safety integrity) prawdopodobieństwo, że system satysfakcjonująco wykona wskazaną funkcję bezpieczeństwa we wszystkich wyspecyfikowanych warunkach w zadanym okresie czasu
nienaruszalność bezpieczeństwa oprogramowania	(ang. software safety integrity) część nienaruszalności bezpieczeństwa powiązana z systematycznymi uszkodzeniami, których źródłem jest oprogramowanie
niezdatność	(ang. fault) nieprawidłowa okoliczność, która może doprowadzić do redukcji bądź utraty zdolności wykonywania pożądanej funkcji przez jednostkę funkcjonalną

niezdatność fizyczna	degradacja fizycznych komponentów systemu zachodząca w trakcie jego pracy mogąca skutkować wystąpieniem uszkodzenia
niezdatność projektowa	wada systemu obecna przy jego wdrożeniu lub wprowadzona w trakcie aktualizacji, która może skutkować wystąpieniem uszkodzenia
ogólna funkcja bezpieczeństwa	(ang. overall safety function) ogół środków mających na celu osiągnięcie bądź utrzymanie stanu bezpiecznego w odniesieniu do konkretnego ryzyka
ogólny cykl życia bezpieczeństwa	(ang. Overall safety lifecycle) ogół działań mających na celu zapewnienie bezpieczeństwa systemu w trakcie całego cyklu użytkowania tego systemu
poziom nienaruszalności bezpieczeństwa, SIL	(ang. safety integrity level) jeden z czterech poziomów odpowiadających określonemu zakresowi wartości nienaruszalności bezpieczeństwa, gdzie poziom 4 oznacza najwyższy poziom nienaruszalności bezpieczeństwa, a poziom 1 najniższy
przywrócenie usługi	przejście od stanu niepoprawnego dostarczania usługi do poprawnego dostarczania usługi
ryzyko	kombinacja prawdopodobieństwa wystąpienia szkody i ciężkości szkody
ryzyko uspołecznione	ryzyko powstałe gdy na skutek pojedynczego niebezpiecznego zdarzenia może zginąć wiele osób. Nazwa ryzyka związana jest z faktem, że wystąpienie takiego zdarzenia z dużym prawdopodobieństwem wywoła reakcję społeczną oraz polityczną
scenariusz zagrożenia	zestaw zależności czasowych opisujących w jakich warunkach wystąpienie akcji zadanych zdarzeń umożliwia materializację zagrożenia. Pojedynczemu MZP może być przypisany jeden bądź więcej scenariuszy zagrożeń
sprzętowa nienaruszalność bezpieczeństwa	(ang. hardware safety integrity) część nienaruszalności bezpieczeństwa związana z niebezpiecznymi losowymi uszkodzeniami sprzętu
stanu bezpieczny	stan o zapewnionym bezpieczeństwie
system	zestaw komponentów łącznie z zależnościami pomiędzy nimi, zaprojektowany celem dostarczania określonej usługi

system E/E/PE związany z bezpieczeństwem	(ang. safety-related E/E/PE system) system E/E/PE, który wypełnia funkcje bezpieczeństwa oraz którego działanie ma wpływ na poziom integralności bezpieczeństwa rozpatrywanych funkcji
systematyczna nienaruszalność bezpieczeństwa	(ang. systematic safety integrity) część nienaruszalności bezpieczeństwa związana z niebezpiecznymi uszkodzeniami systematycznymi
szkoda	obrażenie fizyczne bądź uszczerbek na zdrowiu ludzi bądź zniszczenia mienia bądź w środowisku
T-graf	skierowany graf którego wierzchołki opisane są etykietami a krawędzie parą liczba rzeczywista + relacja (‘>’, ‘≥’), reprezentuje zestaw zależności czasowych
TREM	(Timing Requirement sElection Method) metoda uzyskiwania czasowych wymagań bezpieczeństwa wobec analizowanego systemu
uszkodzenie	ustanie zdolności dostarczania pożądanej funkcji przez jednostkę funkcjonalną bądź działanie jednostki funkcjonalnej w jakikolwiek sposób różny od pożądanego
uszkodzenie bezpieczne	uszkodzenie systemu bądź jego części, które nie ma możliwości przyczynić się do wystąpienia niebezpiecznego zdarzenia oraz do ograniczenia możliwości reakcji systemu na takowe zdarzenie
uszkodzenie niebezpieczne	uszkodzenie systemu bądź jego części, które może skutkować wystąpieniem niebezpiecznego zdarzenia bądź ograniczyć możliwość reakcji systemu na takowe zdarzenie
uszkodzenie systematyczne	(ang. systematic failure) uszkodzenie powiązane w sposób deterministyczny z pewną przyczyną, które może być wyeliminowane poprzez zmianę projektu, procesu produkcji, procedur operacyjnych, dokumentacji lub innych istotnych czynników
uśpiona niezdatność	niezdatność nie powodująca w danym momencie błędu w systemie (może do błędu doprowadzić w przyszłości)
zagrożenie	potencjalne źródło szkody
zbiór cięć minimalnych (MCS)	patrz minimalny zbiór przyczyn
zdarzenie kontrolowalne	zdarzenie proste, których czas wystąpienia można kontrolować

zdarzenie nieobserwowalne	zdarzenie proste, które zachodzą niezaobserwowane
zdarzenie nierozwinięte	zdarzenie proste, które nie zostało rozwinięte dalej z powodu braku informacji bądź ponieważ konsekwencje tego zdarzenie są nieistotne
zdarzenie obserwowalne	zdarzenia inne niż kontrolowalne oraz przewidywalne, których wystąpienie da się odnotować
zdarzenie podstawowe	zdarzenie proste, które nie wymaga dalszego rozwijania (osiągnięto zakładaną rozdzielczość)
zdarzenie pośrednie	w drzewach niezdatności to zdarzenie posiadające podelementy (bramkę). Zdarzenie pośrednie jest zdarzeniem wyjściowym jakiejś bramki (zdarzenie szczytowe jest specyficznym przypadkiem zdarzenia pośredniego)
zdarzenie proste	zdarzenie w drzewie niezdatności, którego bezpośrednich przyczyn nie dociekano (nie jest zdarzeniem wyjściowym żadnej z bramek)
zdarzenie przewidywalne	zdarzenie elementarne inne niż kontrolowalne, o których wystąpieniu wiadomo z wyprzedzeniem
zdarzenie rozwinięte	w drzewach niezdatności to zdarzenie rozwinięte dalej w osobnym drzewie niezdatności
zdarzenie warunkujące	zdarzenie proste używane do określania warunków lub ograniczeń dotyczących dowolnej bramki. Używane głównie z bramką warunkową
zdarzenie zewnętrzne	zdarzenie proste, które wystąpienie jest zakładane podczas normalnej pracy systemu, ale które może przyczynić się do wystąpienia niezdatności

D.2. Wykaz oznaczeń

ϕ funkcja $E \to \mathbb{P}(A)$, która dla zadanego zdarzenia zwraca wszystkie akcje tego zdarzenia. Może być etykietowana wybraną funkcją *Time* i wtedy zwraca akcje danego zdarzenia zachodzące w zachowaniu systemu opisanym zadaną funkcją *Time*

$\overline{}$ operator $\{>, \geq\} \to \{>, \geq\}$ negacji symbolu relacji

$\overline{}$ operator $E \to E$ negacji krawędzi

$\perp$ wyróżniona akcja "cicha" trwająca przez cały okres życia systemu

$\prec_c$ relacja przyczynowości na zbiorze $(T \times T)$. Jest to asymetryczna i przechodnia relacja porządku częściowego

$\prec_h$ relacja przyczynowości "head". Oznacza ona, że akcja będąca drugim argumentem relacji jest spowodowana przez początek akcji będącej pierwszym argumentem

$\prec_i$ relacja przyczynowości "interior". Oznacza ona, że akcja będąca drugim argumentem relacji jest spowodowana przez koniec akcji będącej pierwszym argumentem

$\prec_t$ relacja uporządkowania czasowego na zbiorze $(Time \times Time)$

$<$ relacja mniejszości na zbiorze $(E \times E)$. Jest to relacja przeciwzwrotna, asymetryczna i przechodnia

$\leq$ relacja mniejszości bądź równości na zbiorze $(E \times E)$. Jest to relacja zwrotna, asymetryczna i przechodnia

$=$ relacja równości na zbiorze $(E \times E)$

$=_c$ relacja równości przyczynowej na zbiorze $(T \times T)$. Zachodzi pomiędzy tranzycjami o takich samych przyczynach

$=_t$ relacja równości czasowej na zbiorze $(Time \times Time)$

A zbiór *akcji* – zdarzeń opatrzonych etykietami (różne etykiety umożliwiają rozróżnienie różnych wystąpień tego samego zdarzenia) oraz wyróżnionej akcji "cichej" (oznaczonej $\perp$) trwającej przez cały okres życia systemu. Akcje są oznaczane przez *x*, *y*, *z*

B wartości logiczne logiki boolowskiej

C zbiór zdarzeń *kontrolowalnych* (podzbiór *E*; dopełnienie *U*)

E zbiór *zdarzeń*, którego elementy oznaczamy jako X, Y, Z

edges funkcja zwracająca zbiór krawędzi grafu

enabling_condition wyrażenie opisujące zależności czasowe pomiędzy akcjami zawartymi w argumentach

end funkcja częściowa $A \to \mathbb{R}$, która dla danej akcji i zadanej funkcji *Time* zwraca czas wystąpienia końcowej tranzycji akcji

F zbiór krawędzi grafów

FT zbiór drzew niezdatności

L zbiór *etykiet* używanych do identyfikacji wystąpień zdarzeń; etykiety są oznaczane przez *l*, *m*, *n*

M funkcja semantyczna definiująca bramkę drzewa niezdatności

mk-action funkcja $(Time \times Time) \to A$ umożliwiająca definiowanie akcji na podstawie dwóch tranzycji i czasów ich wystąpień

MZP zbiór *T-grafów* składający się na pojedynczy Minimalny Zbiór Przyczyn

N zbiór zdarzeń *nieobserwowalnych* (podzbiór *U*, rozłączny z *O* oraz *P*)

O zbiór zdarzeń *obserwowalnych* (podzbiór *U*, rozłączny z *P* oraz *N*)

P zbiór zdarzeń *przewidywalnych* (podzbiór *U*, rozłączny z *O* oraz *N*)

$\mathbb{P}(\cdot)$ zbiór potęgowy zbioru podanego w argumencie (zbiór wszystkich podzbiorów tego zbioru, łącznie z nim samym oraz zbiorem pustym)

PE predykat charakterystyczny zdarzenia, określający odpowiadające danemu zdarzeniu stany wybranych elementów systemu (dla zdarzenia prostego stan pojedynczego elementu)

$\mathbb{R}$ zbiór liczb rzeczywistych

start funkcja częściowa $A \to \mathbb{R}$, która dla danej akcji i zadanej funkcji *Time* zwraca czas wystąpienia początkowej tranzycji akcji

T zbiór *tranzycji* będących początkami bądź końcami wybranej akcji. Tranzycje oznaczane są przez symbol akcji wraz z odpowiednim indeksem: $_s$ (początek akcji) lub $_e$ (koniec akcji)

T-graf cykliczny graf skierowany o funkcji wagowej w (reprezentowany przez uporządkowaną parę: $T - graf \stackrel{\text{def}}{=} \big(vertexes(T - graf), edges(T - graf)\big)$)

Time funkcja częściowa $T \to \mathbb{R}$ przypisująca tranzycjom czas ich wystąpienia, może być interpretowana jako zbiór par (w, r), gdzie $r \in \mathbb{R}$, $w \in T$ i $Time(w) = r$. Różne funkcje *Time* reprezentują różne zachowania opisywanego systemu.

U zbiór zdarzeń *niekontrolowalnych* (podzbiór *E*; dopełnienie *C*; suma *P*, *O* oraz *N*)

$vertexes$ funkcja zwracająca zbiór wierzchołków grafu

w funkcja wagowa $F \to \{\mathbb{R} \times \{>, \geq\}\}$ przypisująca krawędziom grafu wagi składające się z liczby rzeczywistej i symbolu relacji