TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Aplikace řešící spolehlivost systému metodou stromu poruchových stavů (FTA)

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Vlastimil Fengl**
Vedoucí práce: Ing.Josef Chudoba, Ph.D.

Liberec 2018/2019

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Application for system reliabilty evaluation using fault tree analysis

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information technology

Author: **Vlastimil Fengl**
Supervisor: Ing.Josef Chudoba, Ph.D.

Liberec 2018/2019

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2018/2019**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Vlastimil Fengl**
Osobní číslo:	**M15000014**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace řešící spolehlivost systémů metodou stromu poruchových stavů (FTA)**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Seznamte se s principy metody FTA využívané v teorii spolehlivosti a rizik, porovnejte existující SW
2. Vytvořte algoritmus tvorby stromu poruchových stavů ze vstupních dat
3. Vytvořte SW řešící aplikaci, grafické rozhraní a grafické výstupy
4. Ověřte správnost řešení vytvořené aplikace

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Fuchs P., Vališ D. Chudoba J., Kamenický J., Zajíček J., Řízení spolehlivosti, učební text, Technická univerzita v Liberci 2006
2. ČSN EN 61025 (010676) Analýza stromu poruchových stavů (FTA), Česká technická norma, 2007
3. Vesely W. E., Goldberg F. F., Roberts N. H., Haasl D. F., Fault tree handbook, U. S. Nuclear Regulatory Commission, 1981, NUREG-0492 https://www.nrc.gov/docs/ML1007/ML100780465.pdf
4. www.weibull.com
5. Knuth D. E., Umění programování, Computer press, Praha 2010, ISBN 978-80-2512-898-5

Vedoucí bakalářské práce: **Ing. Josef Chudoba, Ph.D.**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **3. září 2018**

Termín odevzdání bakalářské práce: **30. dubna 2019**

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 3. září 2018

Abstrakt

Tato bakalářská práce se zabývá metodou stromu poruchových stavů a vytvoření aplikace, která tuto metodu implementuje. Jde o metodu používanou pro ohodnocení spolehlivosti libovolného systému. Práce se zaměřuje na popis a postup metody stromu poruchových stavů. Jsou zde uvedeny krátké rešerše na již existující aplikace. Součástí práce je vytvoření aplikace, která metodu stromu poruchových stavů využívá. Je zde uveden vývoj této aplikace, problémy při vývoji a jejich řešení. V závěru práce jsou uvedeny příklady budoucího možného rozšíření aplikace.

Klíčová slova:

FTA, strom poruchových stavů, binární rozhodovací diagram, BDD, software, Java

Abstract

This bachelor thesis deals with the fault tree analysis and the creation of an application that implements this method. This method is used to evaluate the reliability of any system. The thesis focuses on the description and process of the fault tree analysis. There are short reviews of existing applications. Part of the thesis deals with creation of an application that uses the fault tree analysis, problems and their solution during the development. In the end of the thesis are given some examples of future possible extensions to the application.

Keywords:

FTA, fault tree analysis, binary decision diagram, BDD, software, Java

Poděkování

Chtěl bych poděkovat vedoucímu práce za veškerou pomoc s vypracováním bakalářské práce. Dále bych chtěl poděkovat své přítelkyni, že měla pevné nervy a velmi mi pomohla při vytváření této práce. Nakonec bych chtěl poděkovat svému vedoucímu v práci, který měl pochopení a nechával mi dost času na vypracovávání této práce.

Obsah

Seznam zkratek 10

1 Úvod **11**

2 FTA - Analýza stromu poruchových stavů **12**
2.1 Úvod do FTA 12
2.2 Cíle a použití FTA 12
2.3 Termíny a definice v FTA 13
2.4 Obecný popis FTA 14
2.5 Používané značky v FTA 15
2.6 Postup a vyhodnocení FTA 16

3 Rešerše vybraných SW pro FTA **25**
3.1 Program ALD Fault Tree Analyser 25
3.2 Program ITEM Toolkit 26
3.3 Program TopEvent FTA 26
3.4 Program EMFTA 27

4 Vývoj aplikace **28**
4.1 Požadavky na aplikaci 28
4.2 Použitá technologie 28
4.3 Datové struktury 29
4.4 Řešené problémy v aplikaci 31

5 Testování aplikace **42**
5.1 První testovací strom poruch 42
5.2 Druhý testovací strom poruch 44

6 Popis aplikace **47**
6.1 Hlavní okno programu 47
6.2 Okno úpravy komponenty 50
6.3 Okno výsledků analýzy 52
6.4 Generátor náhodného stromu do CSV souboru 52

7 Závěr práce **53**

Příloha **56**

A Obrázky **56**
A.1 Prostředí programu EMFTA 56
A.2 Okno výsledků 57
A.3 Testovací stromy poruch 59

B Ukázky kódu **61**
B.1 Vyhledání základních událostí pomocí hledání do šířky 61
B.2 Ukázka metody *textModify(String text)* ve třídě *PrimEventPane* . . . 62
B.3 Metoda pro ukládání strom poruch do souboru ve třídě *ObjectFileOperator* 63

C Návod k aplikaci **64**
C.1 Instalace 64
C.2 Popis uživatelského rozhraní 64

D Obsah CD **68**

Seznam obrázků

2.1 Ukázka množiny všech minimálních kritických řezů 18
2.2 Příklad stromu poruchových stavů 19
2.3 Logické hradlo AND 21
2.4 Logické hradlo OR 21
2.5 Majoritní hradlo (hradlo M/N) 22
2.6 Strom poruchových stavů pro kvantitativní výpočet 22

3.1 Program ALD Fault Tree Analyser 25
3.2 Program TopEvent FTA 2017 27

4.1 Část binárního rozhodovacího diagramu 38
4.2 Seřazený binárního rozhodovací diagram funkce $f = ac + bc$ 39
4.3 Sjednocení stejných listů diagramu 39
4.4 Nahrazení uzlu BDD 39
4.5 Sjednocení duplicitních uzlů BDD 40
4.6 Redukovaný BDD funkce $f = ac + bc$ 40
4.7 Postup pro získání disjunktní formy booleovské funkce z BDD 40

5.1 Minimální kritický řez č. 1 43
5.2 Minimální kritický řez č. 2 43
5.3 Výsledek kvantitativní analýzy pomocí aplikace 44
5.4 Výsledek kvalitativní analýzy pomocí aplikace pro druhý test. strom . 45

6.1 Hlavní okno aplikace 47
6.2 Kontextové menu komponenty 49
6.3 Okno pro zadání hodnoty pravděpodobnosti 51
6.4 Okno úpravy komponenty 51

A.1 Program EMFTA 56
A.2 Po kvalitativní analýze stromu 57
A.3 Neopravitelný systém po kvantitativní analýze stromu 57
A.4 Opravitelný systém po kvantitativní analýze stromu 58
A.5 Další opravitelný systém po kvantitativní analýze stromu 58
A.6 První testovací strom poruch 59
A.7 Druhý testovací strom poruch 60

C.1 Kontextové menu komponenty 65

Seznam zkratek

FTA	Fault tree analysis
MKR	Minimální kritický řez
SW	Software
RIA	Rich Internet Application
AADL	Architecture analysis and design language

1 Úvod

Současná doba klade velký důraz na uspokojování potřeb zákazníků. S tím jde ruku v ruce i zvyšování výroby a kvality produktů, které si zákazníci přejí. To sebou nese ale i důraz na zvyšování bezpečnosti těchto produktů a pro firmy i snahu na snižování nákladů na výrobu. Existují i odvětví průmyslu, kde je žádoucí znalost spolehlivosti jejich produktů či velikost dopadu nežádoucích faktorů, které tyto produkty mohou ovlivnit, dříve než tento produkt vytvoří.

Snižování nákladů na výrobu při zachování kvality produktu lze dosáhnout minimalizací nežádoucích stavů a chyb ve výrobním procesu. Spolehlivost produktu lze zase dosáhnout eliminací vysoké poruchovosti prvků nebo úpravou návrhu produktu. Pro zapsání čí odhalení těchto problémů a vyhodnocení jejich dopadu je potřeba analytický nástroj, který dokáže tyto problémy přehledně zaznamenat, vyhodnotit a případně umožnit nalezení řešení.

Jedním z těchto nástrojů, které dokáží odhalit kritická místa, je metoda analýzy stromu poruchových stavů (Fault Tree Analysis). Tento nástroj se běžně využívá například v letectví, energetice nebo i v armádě. Tato metoda je podrobněji popsána v kapitole 2 této práce.

V kapitolách 2.3 až 2.5 lze nalézt obecný popis metody stromu poruchových stavů. Kapitola 2.6 popisuje způsob sestavení analýzy a způsoby jejího vyhodnocení. Dále v kapitole 3 jsou popsány několik existujících programů a zkušenosti s jejich použitím. vytvořil jsem aplikaci, která řeší spolehlivost systému pomocí této analýzy. Kapitola 4 popisuje vývoj aplikace v rámci práce a problémy s jejich řešením, které nastaly během vývoje aplikace.

2 FTA - Analýza stromu poruchových stavů

2.1 Úvod do FTA

Metoda analýzy stromu poruchových stavů (Fault tree analysis - FTA) byla vymyšlena roku 1961 panem H. A. Watsonem ve firmě Bell Telephone Laboratories na zakázku pro Letectvo Spojených států amerických. Zakázka se týkala posouzení odpalovacího řídícího systému pro balistickou střelu Minuteman I. Několik článků, které vysvětlovaly výhody metody FTA, bylo prezentováno na Symposiu bezpečnosti v Seattlu v roce 1965. Prezentace těchto článků je označována za počátek širokého zájmu ve využívání metody FTA jako nástroje pro bezpečnost a spolehlivost velmi složitých systémů jako například jaderný reaktory.[1]

Analýza stromu poruchových stavů je metoda pro identifikaci a analýzu faktorů a podmínek, které způsobují nebo by potenciálně mohly způsobovat výskyt či přispívat k výskytu předem definované takzvané vrcholové události. Touto událostí bývá obvykle zhoršení fungování nebo úplné selhání systému či snížení bezpečnosti nebo zhoršení jiných atributů důležitých pro provoz.[2]

2.2 Cíle a použití FTA

Analýza metodou stromu poruchových stavů je vhodná k analýze systémů skládající se z několika funkčně souvisejících nebo závislých podsystémů. Běžné využití tato metoda nachází například při projektování jaderných elektráren, dopravních a komunikačních systémů, chemických či jiných průmyslových procesů, zdravotnických systémů.[2]

Tato analýza se často používá jako nástroj:

- „k analýze systému, určení jeho bezporuchovosti, identifikaci hlavních přispěvatelů k jeho poruchovosti a k hodnocení změn návrhu.“[2]
- „jako pomoc při úsilí vynaloženém při pravděpodobnostním posuzování rizika.“[2]
- „ke zkoumání systému v průběhu jeho vývoje a předvídání a zabránění vzniku či zmírnění následků potenciální příčiny (příčin) nežádoucí vrcholové události.“[2]

Analýzu FTA je možné využít ve všech etapách návrhu produktů jako analytický nástroj pro identifikaci možných návrhových problémů, dokonce i v časných etapách, kdy nejsou úplné informace o podrobnostech návrhu. Během analýzy FTA se také identifikují možné problémy, které mohou mít původ v namáhání produktu prostředím nebo provozem, ve fyzickém návrhu produktu či v závadách ve výrobních procesech, v provozních nebo údržbářských postupech. Analýzu lze také provádět nezávisle na jiných analýzách bezporuchovosti nebo paralelně s nimi.[2]

Analýza FTA se snaží rozpoznat příčiny nebo kombinace příčin, které vedou k vrcholové události. Dalším cílem analýzy FTA je vyhledání události nebo kombinace událostí, které s největší pravděpodobností jsou příčinou vzniku vrcholové události. Poslední úlohou je vypočítání pravděpodobnosti výskytu událostí.[2]

2.3 Termíny a definice v FTA

výstup (outcome)	Výstup je výsledek dané události nebo jiného vstupu. Výstupem může být událost nebo stav.[2]
vrcholový výstup (top outcome)	Výstup, který je zkoumán sestavením stromu poruchových stavů.[2]
konečná událost (final event)	Událost jejíž výstup je konečným výsledek všech vstupních událostí nebo stavů.[2]
vrcholová událost (top event)	Vrcholová událost je událost, která je předmětem zájmu celé analýzy. Je to výchozí bod umístěný na vrcholku celého stromu analýzy. Lze jí též označit za „konečnou událost (final event)“.[2]
hradlo (gate)	Značka používaná ke zobrazení vazby mezi vstupy a výstupní událostí. Značka hradla definuje logický vztah mezi vstupními událostmi, aby došlo k výstupní události.[2]
kritický řez (cut set)	Množina událostí, které způsobují při současném výskytu výskyt vrcholové události.[2]
minimální kritický řez (minimal cut set)	Konečná množina základních, dále nerozvíjených událostí, které se nutně musí vyskytnout, aby způsobily výskyt vrcholové události.[2]
událost (event)	Výskyt určité podmínky nebo děje.[2]
základní událost (basic event)	Událost, kterou nelze dále rozvíjet.[2]

primární událost (primary event)	Událost nacházející se na nejnižší úrovni stromu poruchových stavů.[2]
mezilehlá událost (intermediate event)	Událost, která není ani vrcholová, ani primární. Většinou je výsledkem jedné nebo více primárních událostí nebo jiných mezilehlých událostí.[2]
nerozvíjená událost (undeveloped event)	Událost neobsahující vstupní události. V analýze bývá takto označována událost buď z nedostatku podrobnějších informací, nebo je tato událost rozvíjená v jiné analýze a v aktuální analýze je označena jako nerozvíjená.[2]
jednobodová porucha (single point failure)	Událost, která způsobí vrcholovou událost bez ohledu na ostatní události.[2]
opakovaná událost (repeated event)	Událost, která je vstupem pro jiné události vyšší úrovně.[2]

2.4 Obecný popis FTA

Grafické znázornění analýzy FTA je strom poruchových stavů. Jde o organizovanou reprezentaci podmínek nebo jiných faktorů, které způsobují nebo přispívají k výskytu hledané vrcholové události. Strom poruchových stavů je zobrazován jako diagram, který vyobrazuje logické vztahy mezi vrcholovou událostí a základními událostmi, které mohou způsobovat vrcholovou událost. Strom poruch vyobrazí vývoj poruchy v systému a odhalí vazby mezi jednotlivými prvky systému.[3]

2.4.1 Součásti diagramu FTA

Diagram FTA se skládá z těchto součástí:

Hradla

Hradla znázorňují logické vztahy mezi vstupními událostmi a výstupními událostmi. Dále se dělí na statická a dynamická.[3]

- **Statická hradla** – výstup z těchto hradel nezávisí na pořadí výskytu na vstupu.
- **Dynamická hradla** – výstup z těchto hradel je závislý na pořadí výskytu na vstupu.

Události

Události se nachází na nejnižší úrovni vstupů ve stromu poruchových stavů. Do stromu poruchových stavů je nutné zahrnout všechny relevantní události včetně vlivu prostředí a ostatní podmínky namáhání, kterým muže být objekt vystaven.[3]

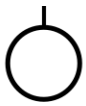
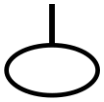
Je nutné zahrnout i takové události, které jsou během provozu možné, ale i mimo specifikaci návrhu. Události, které jsou v analýze uvažovány, ale vyloučeny z další analýzy jako nepoužitelné, mají být dokumentovány ve zprávě analýzy, ale už nemají být zahrnuty v konečném stromu poruchových stavů.[2]

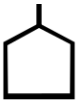
Ostatní

Ostatní grafické součásti diagramu FTA jsou:

- spojovací čáry
- popis mezilehlých událostí
- značky transferu dovnitř a ven
- značky primárních událostí

2.5 Používané značky v FTA

	Základní událost **Basic event**	Událost nacházející se na nejnižší úrovni stromu. Pro tuto událost jsou k dispozici informace o bezporuchovosti nebo pravděpodobnosti výskytu.[2]
	Nerozvíjená událost **Undevelopment event**	Primární událost, která představuje dosud nerozvíjenou část systému.[2]
	Neaktivní událost **Nonactive event**	Primární událost reprezentující neaktivní poruchu.[2]
	Podmínková událost **Conditional event**	Událost, která je podmínkou výskytu další události. Musí nastat obě události, aby nastal výstup.[2]
	Transfer **Transfer**	Hradlo, které naznačuje, že je tato část systému rozvíjena na jiné straně nebo v jiné části diagramu.[2]

	Hradlo AND **AND Gate**	Výstupní událost hradla nastane pouze tehdy, jestliže nastanou všechny vstupní události.[2]
	Majoritní hradlo **Major Gate**	Výstupní událost hradla nastane pouze tehdy, jestliže nastane m nebo více vstupních událostí z celkového počtu n vstupních událostí.[2]
	Hradlo OR **OR Gate**	Výstupní událost hradla nastane, jestliže nastane jakákoliv ze vstupních událostí.[2]
	Hradlo XOR **XOR Gate**	Výstupní událost hradla nastane, jestliže nastane právě jedna z událostí na vstupu hradla.[2]
	Hradlo PAND **PAND Gate**	Výstupní událost hradla nastane, jestliže nastanou všechny vstupní události v pořadí zleva doprava.[2]
	Hradlo INHIBIT **INHIBIT Gate**	Výstupní událost hradla nastane jen tehdy, jestliže nastanou obě vstupní události, z nichž jedna je podmínková událost.[2]
	Negace **NEGATION**	Výstupní událost nastane, jestliže nenastane vstupní událost.[2]
	Přepínač událostí **SWITCH event**	Výstupní událost, která se buď jistě stane nebo stala.[2]

2.6 Postup a vyhodnocení FTA

Metoda stromu poruchových stavů je deduktivní metoda (strom se analyzuje od shora směrem dolů) a je zaměřená na přesném určení příčin nebo kombinací příčin, které by mohly vyvolat vrcholovou událost. Samotnou analýzu by měli provádět pracovníci, kteří jsou jak vycvičeni k použití analýzy FTA nebo jiné příslušné techniky modelování bezporuchovosti, tak mají podrobné znalosti analyzovaného systému a jeho návrhových rysů o provozu. Analytik by měl dokázat vyhodnotit vlivy potencionálních způsobů poruch a logicky jednoznačně určit možné příčiny nestandardního chování.[2]

2.6.1 Postup sestavení FTA

Před zahájením analýzy FTA je potřeba celý systém vymezit tak, aby mohl být reprezentován jako funkční bloky, kde mohou být určeny technické parametry jednotlivých prvků. Z tohoto důvodu bývá pro analýzu FTA vytvořen tým odborníků, kteří s analyzovaným systémem pracují, mají znalosti či zkušenosti s jednotlivými částmi analyzovaného systému nebo budou aplikovat analýzu FTA. Z těchto znalostí se pak hledají všechny možné způsoby poruch a případná nedostatečná znalost systému neumožní odhalit všechny možné poruchy. Dále je nutné seznámit se se všemi souvislostmi a vazbami mezi jednotlivými bloky systému, u kterých může docházet k poruchám, a je nutné je rozpoznat a zaznamenat do stromu.

Pak přichází definice vrcholové události. Tato událost bude stav(porucha), který je v systému nežádoucí a pro který se budou hledat všechny možné příčiny jeho vzniku. Při vytváření stromu se postupuje od vrcholové události, přes mezilehlé události, až k primárním událostem, které jsou prvotní příčinou vzniku poruchy.[3]

Dalším krokem je sestavení stromu poruchových stavů. Pro sestavení stromu se používají výše definované značky z kapitoly 2.5 Používané značky v FTA. Existují dvě skupiny značek, události a hradla. Vztahy mezi událostmi jsou definovány pomocí značek hradel. V konečném stádiu vývoje vznikne diagram, ve kterém jsou všechny události spojeny hradly a každé hradlo má jednu výstupní a jednu či více vstupních událostí.[4]

Po sestavení stromu poruch se může přejít k samotné analýze tohoto stromu. Existují dva způsoby vyhodnocení stromu, buď kvalitativní nebo kvantitativní analýza. Kvalitativní analýza po vyhodnocení vrací množinu základních událostí, které při současném vzniku vedou ke vzniku vrcholové události. U kvantitativní analýzy stromu dojde k pravděpodobnostnímu ohodnocení vzniku vrcholové události na základě pravděpodobnostních ohodnocení základních událostí.

Postup při FTA

- vymezení rozsahu analýzy
- seznámení se s návrhem, funkcemi a provozem systému
- definování vrcholové události
- vytvoření stromu poruchových stavů
- analýza logiky stromu poruchových stavů (kvalitativní nebo kvantitativní)
- vytvoření zprávy o výsledcích analýzy
- posouzení zlepšení bezporuchovosti systému

2.6.2 Kvalitativní analýza FTA

Cílem této analýzy je nalezení všech možných kombinací faktorů, podmínek prostředí, chyb lidského faktoru a poruch komponent systému, které by mohly vést ke vzniku vrcholové události. Přesněji analýza umožňuje nalézt množinu všech minimálních kritických řezů.[4]

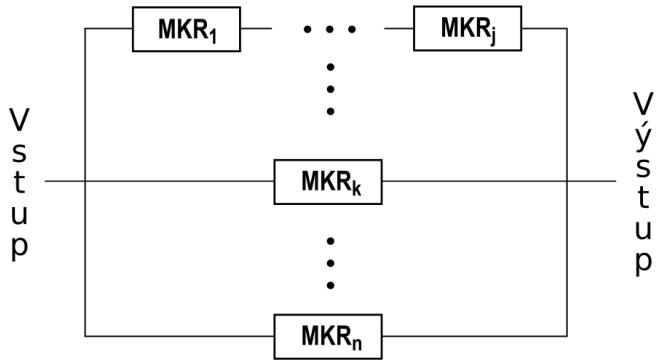
Kritický řez stromu poruchových stavů

Kritický řez je taková množina základních, dále nerozvíjených událostí, které, pokud nastanou současně, vedou ke vzniku vrcholové události.

Minimální kritický řez stromu poruchových stavů

Minimální kritický řez (dále jen MKR) je taková množina elementárních událostí, která je sama kritickým řezem, ale současně žádná její vlastní podmnožina kritickým řezem není.

Jestliže je množina všech minimálních kritických řezů MKR_i (kde i = 0, 1, ... j, ... k, ... n) známa, lze logickou strukturu stromu poruchových stavů překreslit na sériově paralelní blokový diagram, kde každá větev diagramu představuje jeden minimální kritický řez (viz Obr. 2.1).[4]



Obrázek 2.1: Ukázka množiny všech minimálních kritických řezů

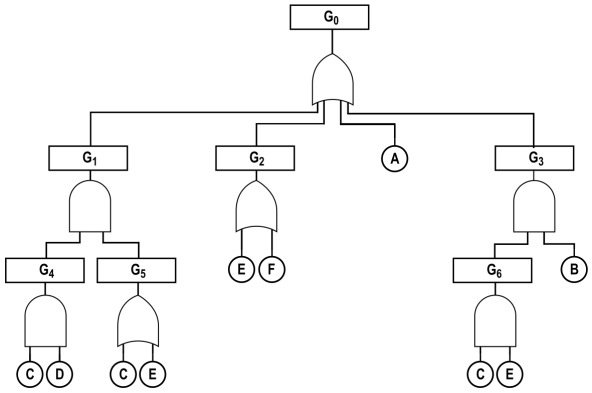
Algoritmus k vyhledání minimálních kritických řezů

Jedním z požadavků při řešení stromu poruchových stavů je nalezení všech minimálních kritických řezů. Nalezení množiny všech minimálních kritických řezů umožní přetvořit logiku všech variant poruchy systému na jednoduchou sériově-paralelní blokovou strukturu, kterou lze řešit standardními výpočty.

Základní metodou pro nalezení všech minimálních kritických řezů je Booleovská redukce, která je založená na popisu logických vazeb vyjádřených stromem poruchových stavů. Tuto metodu lze použít i pokud se objevují stejné události na více místech ve stromě poruchových stavů. Bohužel tuto metodu nelze použít pokud je vrcholová událost časově závislá na jevech nebo na posloupnosti jevů.

Metoda je založená na postupném vyjádření vrcholové události jako kombinace jednotlivých událostí popsaných ve stromě poruchových stavů. V prvním kroku se vyjádří vrcholová událost jako logická kombinace událostí, které bezprostředně mohou způsobovat vrcholovou událost. V dalších krocích se stejným způsobem popisují události na nižších úrovní stromu dokud vrcholová událost není vyjádřena jako logická kombinace všech základních událostí.

Výsledný logický výraz se dále upraví a zjednoduší za pomoci Booleovy algebry tak, aby vyjadřoval prosté sjednocení průniků základních událostí. Tyto průniky pak reprezentují minimální kritické řezy stromu poruchových stavů.[4]



Obrázek 2.2: Příklad stromu poruchových stavů

Celý postup bude podrobněji ukázán na příkladu. Je dán strom poruchových stavů, který je vyobrazen na Obr. 2.2. Nejprve je nutné vyjádřit vrcholovou událost G_0 jako logickou kombinaci bezprostředních příčin této události:

$$G_0 = G_1 + G_2 + A + G_3 \tag{2.1}$$

Dále je potřeba vyjádřit jevy G_1, G_2 a G_3 jako logické kombinace jejich bezprostředních příčin a tento bod opakovat dokud celá rovnice není vyjádřena výhradně jen elementárními jevy:

$$G_0 = (G_4 \cdot G_5) + (E + F) + A + (G_6 \cdot B) \tag{2.2}$$

$$G_0 = ((C \cdot D) \cdot (C + E)) + (E + F) + A + ((C \cdot E) \cdot B) \tag{2.3}$$

Potom výraz upravit tak, aby vyjadřoval prosté sjednocení jevů:

$$G_0 = A + C \cdot C \cdot D + C \cdot D \cdot E + E + F + B \cdot C \cdot E \tag{2.4}$$

Tento výraz lze dále zjednodušit, protože v Boolově algebře platí, že:

$$C \cdot C \cdot D = C \cdot D \tag{2.5}$$

$$C \cdot D + C \cdot D \cdot E = C \cdot D \tag{2.6}$$

$$E + B \cdot C \cdot E = E \tag{2.7}$$

Výsledný logický výraz má po zjednodušení tvar:

$$G_0 = A + C \cdot D + E + F \tag{2.8}$$

Pro ukázkový strom poruchových stavů byly nalezeny čtyři minimální kritické řezy:

$$\sum MKR = \{A\}, \{C \cdot D\}, \{E\}, \{F\} \tag{2.9}$$

Výše uvedený postup je velice jednoduchý a vede k jednoznačnému určení všech minimálních kritických řezů. Avšak při vysokém počtu elementárních jevů je tento postup velmi obtížné ručně zvládnout, proto na řadu přicházejí specializované softwarové produkty.[4]

Hodnocení závažnosti minimálních kritických řezů

Strom poruchových stavů může být kvalitativně posouzen na základě rozboru minimálních kritických řezů podle různých kritérii závažnosti. Jedním z důležitých kritérií závažnosti je *řád řezu*. Řád řezu minimálního kritického řezu udává počet elementárních jevů, které tento řez obsahuje. Minimální kritický řez prvního řádu je většinou závažnější než řezy vyšších řádů, protože takový řez se sestává pouze z jednoho elementárního jevu a při vzniku tohoto jevu dochází i k vrcholové události. U vyšších řádů nastává vrcholová událost jen pokud nastanou všechny elementární jevy řezu současně.[4]

2.6.3 Kvantitativní analýza FTA

Jestliže jsou určeny parametry spolehlivosti elementárních jevů, lze provést kvantitativní analýzu stromu poruchových stavů. Tato analýza umožňuje určení celé řady ukazatelů, které charakterizují vrcholovou událost. Zde je výčet některých ukazatelů:

- pravděpodobnost výskytu vrcholové události v zadaném intervalu provozu systému
- střední doba do prvního nastoupení vrcholové události

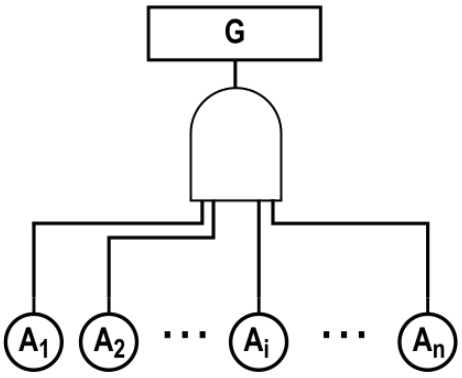
Tato práce se bude dále zabývat jen stanovením pravděpodobnosti výskytu vrcholové události. Všechny metody výpočtů stromu poruchových stavů jsou velmi složité v závislosti na rozvětvenosti stromu. Proto se ve většině případů používají softwarové programy vyvíjené ve specializovaných softwarových firmách zabývající se v oblasti spolehlivostí.[4]

Nejčastěji používané metody při výpočtech stromu poruchových stavů:

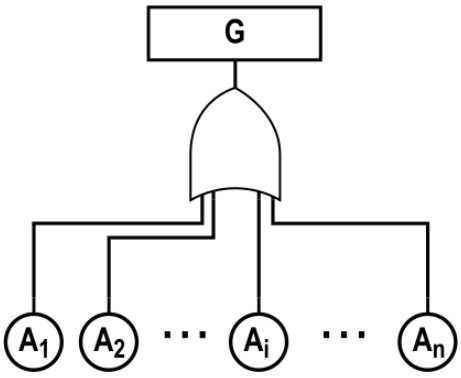
- metoda přímého výpočtu
- metoda minimálních kritických řezů
- simulační metody (Monte Carlo)

Metoda přímého výpočtu

Tato metoda spočívá ve výpočtu pravděpodobnosti vzniku události na hradle, které definuje vztahy mezi elementárními událostmi na vstupu hradla. Tuto metodu lze použít pouze u stromu poruchových stavů, kde se nachází každá elementární událost pouze jednou. Tohoto stavu lze dosáhnout úpravou logického výrazu vrcholové události na disjunktní formu.



Obrázek 2.3: Logické hradlo AND



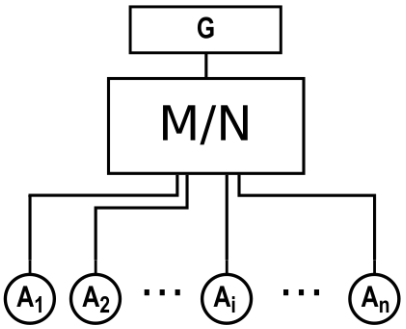
Obrázek 2.4: Logické hradlo OR

Pravděpodobnost události G u hradla AND (viz Obr. 2.3) se vypočítá podle vzorce:

$$P(G) = \prod_{i=1}^{n} P(A_i) \tag{2.10}$$

U hradla OR (viz Obr. 2.4) se pravděpodobnost události G vypočítá:

$$P(G) = 1 - \prod_{i=1}^{n} [1 - P(A_i)] \tag{2.11}$$



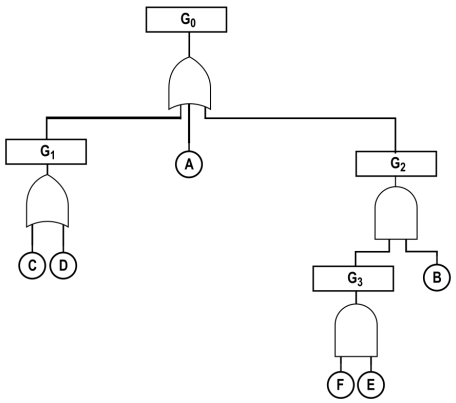
Obrázek 2.5: Majoritní hradlo (hradlo M/N)

Pravděpodobnost události G na hradle M/N (viz Obr. 2.5) se při shodných pravděpodobnostech potomků vypočítá:

$$P(G) = \sum_{i=m}^{n} \binom{n}{i} \cdot P^i \cdot (1-P)^{n-i} \tag{2.12}$$

kde P je pravděpodobnost potomků. Pokud jsou pravděpodobnosti potomků odlišné, vypočítá se jako součet součinů pravděpodobností všech n-tic potomků, kde alespoň m potomků z n jsou porouchané.

Pravděpodobnost vrcholové události stromu poruchových stavů se dopočítá tak, že za pomoci známých vzorců se postupně určí pravděpodobnost jevů od nejnižší úrovně až k vrcholové události. Strom se prochází postupně odspodu po všech hradlech a podle jejich typu se určí pravděpodobnost vzniku jevů, které jsou těmito hradly definovány. Postupnou aplikací vzorců (2.10) a (2.11) se určují pravděpodobnosti všech neelementárních jevů (mezilehlých událostí).[4]



Obrázek 2.6: Strom poruchových stavů pro kvantitativní výpočet

Postup výpočtu pravděpodobnosti vrcholové události bude ukázán na příkladu. Je dán strom poruchových stavů ukázaný na Obr. 2.6. Pravděpodobnosti výskytu chyb jednotlivých elementárních událostí nabývají následujících hodnot:

- Událost A – $P(A) = 0.1$
- Událost B – $P(B) = 0.02$
- Událost C – $P(C) = 0.25$
- Událost D – $P(D) = 0.03$
- Událost E – $P(E) = 0.125$
- Událost F – $P(F) = 0.02$

Nejprve se vypočítá pravděpodobnost na hradlech, jejichž potomci jsou elementární události. V příkladu to jsou hradla G_1 a G_3. Pravděpodobnost výskytu chyby na hradle G_1 se vypočítá pomocí vzorce (2.11) následovně:

$$\begin{aligned} P(G_1) &= 1 - [(1 - P(C)) \cdot (1 - P(D))] \\ P(G_1) &= 1 - [(1 - 0.25) \cdot (1 - 0.03)] \\ P(G_1) &= 0.2725 \end{aligned} \tag{2.13}$$

Obdobně se vypočítá pravděpodobnost výskytu chyby na hradle G_3, ale použije se k tomu vzorec (2.10), protože se jedná o hradlo AND:

$$\begin{aligned} P(G_3) &= P(F) \cdot P(E) \\ P(G_3) &= 0.02 \cdot 0.125 \\ P(G_3) &= 0.0025 \end{aligned} \tag{2.14}$$

Dále je potřeba vypočítat pravděpodobnost výskytu chyby na hradle G_2, aby bylo možno dopočítat pravděpodobnost na vrcholové události G_0. Pro výpočet pravděpodobnosti na hradle G_2 se využije vzorec (2.10):

$$\begin{aligned} P(G_2) &= P(G_3) \cdot P(B) \\ P(G_2) &= 0.0025 \cdot 0.02 \\ P(G_2) &= 0.00005 \end{aligned} \tag{2.15}$$

Pravděpodobnost výskytu vrcholové události G_0 se vypočítá za pomoci vzorce (2.11), protože se jedná o hradlo OR. Výpočet vrcholové události vypadá následovně:

$$\begin{aligned} P(G_0) &= 1 - [(1 - P(G_1)) \cdot (1 - P(A)) \cdot (1 - P(G_2))] \\ P(G_0) &= 1 - [(1 - 0.2725) \cdot (1 - 0.1) \cdot (1 - 0.00005)] \\ P(G_0) &\doteq 0.34528 \end{aligned} \tag{2.16}$$

Vrcholová událost G_0 podle výpočtu v příkladu (2.16) nastane s pravděpodobností $P(G_0) \doteq 0.34528$ (zaokrouhleno na pět desetinných míst).

Metoda minimálních kritických řezů

Předpokladem použití této metody je nalezení množiny všech minimálních kritických řezů stromu poruchových stavů. Pokud množina všech minimálních kritických řezů je známa, může být strom transformován na sériově paralelní blokový diagram, kde každá větev reprezentuje jeden minimální kritický řez. Jestliže se nachází každý elementární jev v tomto blokovém diagramu jen jednou, lze vypočítat pravděpodobnost vrcholové události tak, že vypočítáme nejprve pravděpodobnost jednotlivých větví a pak celého diagramu. Jinak se musí blokový diagram přepsat do logické formy a následně tento výraz upravit do disjunktní formy, ze které lze už vypočítat pravděpodobnost vrcholové události.[4]

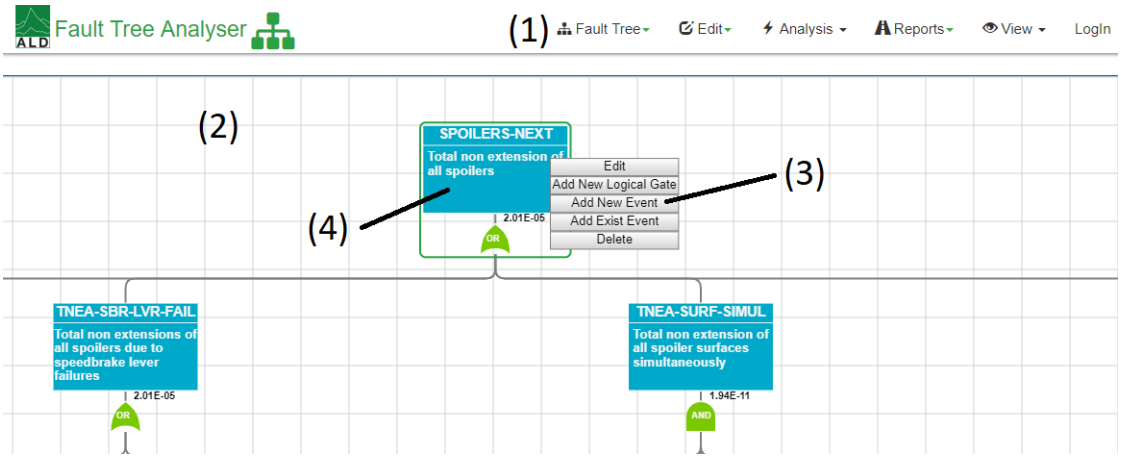
3 Rešerše vybraných SW pro FTA

Tato kapitola se bude zabývat již existujícími softwarovými nástroji pro vytváření a vyhodnocení stromu poruchových stavů. Bude zde krátká zmínka o firmě, která za vývojem daného softwaru stojí. Dále bude obecně popsán jejich produkt a základní popis orientace v prostředí jejich aplikace.

3.1 Program ALD Fault Tree Analyser

Tento nástroj byl vytvořen společností Advanced Logistics Developments. Jedná se o izraelskou společnost a vývojové studio, které od roku 1984 pracuje v oblasti spolehlivosti a její analýzy (Reliability Engineering and Analysis), analýzy bezpečnosti a řízení bezpečnosti, kvality a zajištění kvality (Quality Engineering and Quality Assurance).[5]

Program je napsán jako webová aplikace a je tedy volně dostupný na stránkách `http://www.fault-tree-analysis-software.com/fault-tree-analysis`. Pro jeho využívání je potřeba být zaregistrován. Aplikace umožňuje pouze kvantitativní vyhodnocení stromu. Dále umožňuje uložení/načtení stromu ze souboru, uložit strom jako obrázek, zobrazení duplicitních elementárních jevů, zobrazení listu všech elementárních jevů nebo všech mezilehlých událostí.



Obrázek 3.1: Program ALD Fault Tree Analyser

Prostředí aplikace se skládá ze dvou částí, menu (viz Obr. 3.1 číslo (1)) a pracovní plocha(viz Obr. 3.1 číslo (2)). Jednotlivé komponenty se přidávají pomocí kontextového menu (viz Obr. 3.1 číslo (3)), které je vyvolané pravým kliknutím myši na komponentu, ke které ji chceme připojit. Vzhled událostí v programu je vyobrazen na Obr. 3.1 číslo (4).

3.2 Program ITEM Toolkit

ITEM Toolkit byl vyvinut společností ITEM Software, která sídlí ve Velké Británii. Tato společnost se zabývá vývojem softwarů v oblasti spolehlivosti, analýzy bezpečnosti a vyhodnocení rizik.[6]

ITEM Toolkit je klasická desktopová aplikace, která je zpoplatněná, avšak firma nabízí plně funkční 30denní zkušební verzi. Aplikace se skládá z několika modulů, které je možné si vyzkoušet či koupit zvlášť podle potřeby. FTA modul umožňuje jak kvalitativní tak i kvantitativní vyhodnocení stromu poruch. Dále umožňuje vytvářet více stromů v rámci jednoho projektu, kopírovat události mezi projekty, vygenerovat uživatelsky nadefinované zprávy k projektu a další.[7]

Uživatelské prostředí se skládá z menu, panelu nástrojů (zde lze nalézt jednotlivé komponenty pro vytváření stromu poruch), projektového a systémového okna. Projektové okno zobrazuje jednotlivé projekty, které jsou momentálně otevřené. Projekty se skládají z jednotlivých systémů (analýz), které se v projektu provádí. V systémovém okně se pak vytváří samotný strom poruchových stavů. Jednotlivé komponenty stromu se přidávají jejich výběrem z panelu nástrojů.

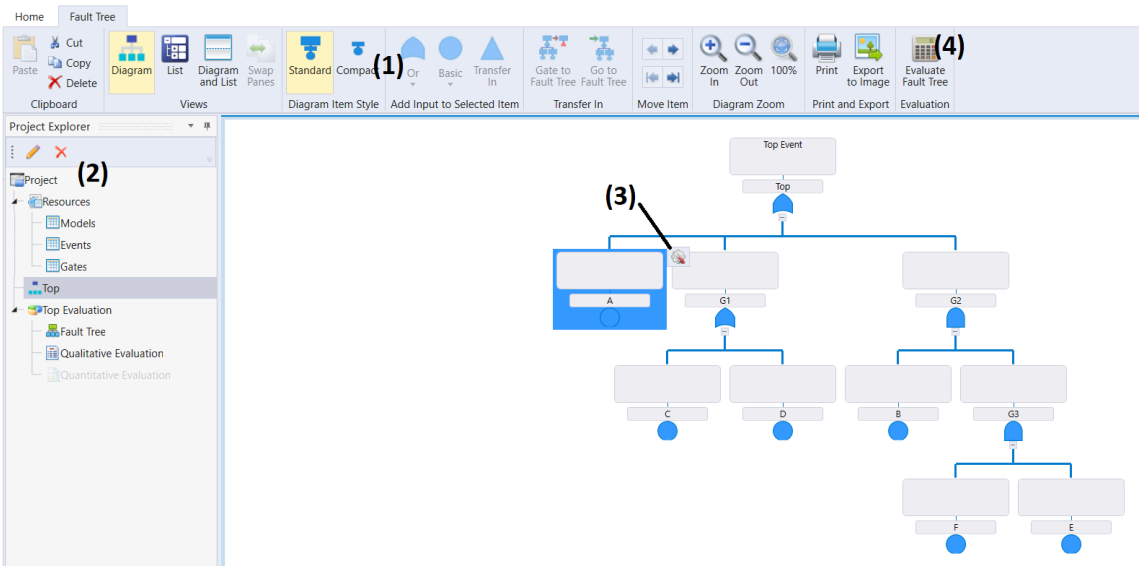
3.3 Program TopEvent FTA

TopEvent FTA je také desktopová aplikace. Firma Reliotech nabízí tento program ve čtyrech verzích (Expert, Starter, Standart, Professional). Verze se mezi sebou liší možným počtem použitých komponent ve stromě poruch a způsoby vyhodnocení stromu[1]. Verze Expert je základní a zadarmo.

Na obrázku 3.2 je vyobrazeno uživatelské prostředí programu. V horní části okna se nachází menu (viz číslo (1) na obrázku 3.2). Pomocí tohoto menu se sestavuje strom poruch tak, že se označí komponenta na pracovní ploše a pak se z menu vybere komponenta, které má být připojena k označené komponentě. V levé části obrazovky se nachází průzkumník projektu, který zobrazuje všechny stromu poruch vytvořené v rámci projektu a výsledky vyhodnocení (viz číslo (2) na obrázku 3.2). Číslo (3) na obrázku 3.2 ukazuje na tlačítko pro nastavení události nebo hradla. Obecně se dá nastavit název a popis komponenty. Pokud je nastavována událost je tam možnost nastavení pravděpodobnosti výskytu poruchy, intenzitu poruchy,

[1] Více informací na adrese https://www.fault-tree-analysis.com/pricing

střední dobu do poruchy a další. Pomocí tlačítka označeného na obrázku 3.2 číslem (4) se spouští vyhodnocení všech stromů poruchových stavů v projektu. Před vyhodnocením se program zeptá jak má strom vyhodnotit, jestli kvalitativně nebo kvalitativně a kvantitativně, a jakým způsobem.



Obrázek 3.2: Program TopEvent FTA 2017

3.4 Program EMFTA

Jedná se o open source[2] program, který je vytvořen jako rozšíření pro existující editor OSATE[3]. OSATE je program postavený na již existujícím editoru Eclipse, který se pro svůj otevřený kód hojně využívá pro modifikace na editor na míru podle potřeby.[9]

Ačkoliv se jedná o volně šiřitelný software, nemůžou ho využívat nezkušení uživatelé jako to bylo u předchozích programů. Nejprve je nutné si nainstalovat OSATE program. Do něj je potřeba doinstalovat rozšíření EMFTA. Návod na instalaci a stažení rozšíření je uvedeno na stránce `https://github.com/cmu-sei/emfta`. Po samotném nastavení programu lze konečně vytvářet strom poruch. Ovládání programu není intuitivní, většinu času se stráví hledáním v manuálu. Předešlé programy oproti tomuto měly snadné ovládání, bylo zřejmé jak vytvořit strom poruch a jak ho vyhodnotit. Ukázka programu EMFTA je na obrázku A.1.

[2]Open source program znamená, že jsou k programu volně dostupné zdrojové kódy a je volně šiřitelný.

[3]OSATE je editor vytvořený pro analýzu a design architektury. Využívá pro své modelování AADL jazyk.

4 Vývoj aplikace

V kapitole Vývoj aplikace bude popsán vývoj grafického editoru pro konstrukci stromu poruchových stavů a jeho vyhodnocení. Níže budou popsány požadavky na systém, technologii použitou pro vývoj aplikace a důležité části kódu programu.

4.1 Požadavky na aplikaci

Systém má být vytvářen jako aplikace s grafickým rozhraním. Měl by umožňovat uživateli používat základní značky (viz kapitola 2.5 Používané značky v FTA) pro grafickou tvorbu stromu poruchových stavů. Dále by systém měl umožňovat nastavení názvu, popisu a pravděpodobnosti prvků, propojování, mazání či duplikování prvků. Také i uložení práce a nahrávaní uložené práce.

Systém by měl umožňovat vyhodnocení stromu jak kvalitativně, tak kvantitativně. Při kvantitativním vyhodnocení stromu by měl systém vyobrazit do grafu, kde je pravděpodobnost závislá na čase. Zadání časových hodnot by mělo být uživateli umožněno.

4.2 Použitá technologie

4.2.1 Java 8 SE

Java je jeden z nejrozšířenějších a nejpoužívanějších programovacích jazyků na světě a to díky přenositelnosti kódu mezi různými platformami. Jedná se o platformy od čteček čipových karet, přes desktopy, vestavěné systémy až k distribuovaným systémům. Jedná se o objektově orientovaný programovací jazyk, který byl vytvořen společností Sun Microsystems v roce 1995 a později dále udržován a rozvíjen společností Oracle.[10]

Program je napsán v tomto programovacím jazyku, protože byla snaha o vytvoření multiplatformní aplikace. Java verze 8 byla vybrána, protože je to poslední verze, která obsahuje knihovny JavaFX. Následující verze už tuto knihovnu nemají a je nutné používat pro vývoj grafického rozhraní knihovny třetích stran.

4.2.2 JavaFX

Jedná se o sadu knihoven pro tvorbu grafického rozhraní v prostředí Java. Slouží především pro vývoj RIA aplikací, což jsou webové aplikace, které mají některé vlastnosti desktopových aplikací.[11]

Byla zde i možnost použít starší knihovnu *Swing*, která je stále používána ve většině desktopových aplikací psaných v jazyce Java. Knihovna *Swing* nebyla vybrána, protože je oproti JavaFX knihovně složitější pro použití v kódu a v knihovně JavaFX se snadněji vytváří vzhled okna programu.

4.2.3 JavaFX Scene Builder

JavaFX Scene Builder je nástroj pro rychlé a snadné vytváření návrhu uživatelského rozhraní JavaFX aplikací. Jednoduchý editor, ve kterém se pomocí „drag & drop“ poskládají jednotlivé uživatelské komponenty na pracovní plochu, upraví se jejich vlastnosti dle potřeb. Výsledkem je pak FXML soubor, který může být použit v JavaFX projektu.[12]

4.2.4 IntelliJ IDEA

Vývojové prostředí, které umožňuje psát aplikace v programovacím jazyce Java, Scala, Kotlin a jiné. Tento produkt byl vytvořen firmou JetBrains. Existuje ve dvou verzích, komunitní a komerční.[13]

Existují i jiné vývojové prostředí (např. NetBeans či BlueJ), ale IntelliJ IDEA byla vybrána z důvodu vynikající orientace v tomto programu a dlouholeté zkušenosti s ním.

4.2.5 Inkscape

Inkscape je grafický editor pro vytváření vektorové grafiky. Je to svobodný a volně šiřitelný program pod licencí GPL. Tento program byl použit na vytvoření obrázků pro tlačítka v programu a také obrázků zde v práci.[14]

4.3 Datové struktury

Strom poruchových stavů je v aplikaci reprezentován jako datová struktura strom skládající se uzlů datového typu *IFTANode*, který je dále v textu popsán. Dále bylo třeba implementovat binární rozhodovací diagram (jeho význam a použití bude vysvětleno v kapitole 4.4.5 Kvantitativní vyhodnocení FTA) a je reprezentován také jako datová struktura strom, ale skládá se z uzlů datového typu *BddNode*.

Dále v této podkapitole budou stručně popsány vybrané datové typy a struktury aplikace, které byly během vývoje vytvořeny.

4.3.1 IFTANode

Rozhraní *IFTANode* popisuje základní společné metody, které musí jednotlivé komponenty stromu poruchových stavů implementovat. Mezi tyto metody patří:

- získání a nastavení názvu komponenty
- získání a nastavení podrobnějšího popisu komponenty
- získání a nastavení pravděpodobnosti výskytu události
- získání a nastavení intenzity poruchy události
- vyhodnocení komponenty
- vyjádření komponenty pomocí booleovské funkce

4.3.2 AEvent

Abstraktní třída *AEvent* reprezentuje v aplikaci libovolnou událost stromu poruchových stavů. Implementuje rozhraní *IFTANode*. Tato třída si ukládá název, popis a pravděpodobnost výskytu komponenty a odkaz na potomka typu *IFTANode*, jehož je předkem. Z této třídy dědí jednotlivé třídy, které reprezentují události stromu poruchových stavů.

4.3.3 AGate

Abstraktní třída *AGate* reprezentuje v aplikaci libovolné hradlo stromu poruchových stavů. Také implementuje rozhraní *IFTANode*. Udržuje si informace o názvu, popisu a pravděpodobnosti výskytu komponenty a odkazy na potomky typu *IFTANode*, jejíž je předkem. Třídy, které reprezentují hradla stromu poruchových stavů, dědí z této třídy.

4.3.4 BddNode

Třída *BddNode* reprezentuje jeden uzel binárního rozhodovacího diagramu. Tato třída má deklarované proměnné:

- znak, který reprezentuje proměnnou v boolovské funkci
- odkazy na dva potomky datového typu *BddNode*

4.3.5 BddTree

Třída *BddTree* má reprezentovat celý binární rozhodovací diagram. Binární rozhodovací diagram se graficky vyjadřuje jako binární strom, třída *BddTree* obsahuje tedy odkaz na kořen tohoto stromu a má implementovaný funkce pro vytvoření takového stromu z booleovské funkce či transformování binárního rozhodovacího diagramu na redukovanou verzi.

4.4 Řešené problémy v aplikaci

4.4.1 Vizualizace FTA komponent v programu

Jednou z podstatných částí programu bylo vymyslet grafické zobrazení stromu poruchových stavů pro uživatele. Bylo třeba vytvořit objekt, který bude zastupovat komponentu ze stromu poruchových stavů, lze graficky vyobrazit, zobrazí i název definovaný uživatelem a bude snadné tento objekt posouvat po pracovní ploše.

Grafické znázornění komponenty

Z knihovny JavaFX byla pro výše zmíněné účely použita třída *Pane*. Jedná se o základní třídu, kterou dědí skoro všechny kontejnery, a lze jí nastavit pozici v nadřazeném kontejneru (v tomto případě je nadřazený kontejner pracovní plocha). Kontejner v JavaFX je komponenta, do které se umisťují komponenty JavaFX jako např. tlačítko, textové pole, obrázek, graf, jiné kontejnery atd. Pro každou značku používanou v FTA (viz kapitola 2.5 Používané značky v FTA) byla vytvořena třída, která dědí z *Pane* a slouží jako kontejner pro jednotlivé části značky, ze kterých se skládá, a pro popisový blok značky. Jednotlivé třídy se v projektu nachází v balíčku *panes*.

Úprava názvu komponenty

Každá komponenta FTA má popisový blok, ve kterém by měl být napsán krátký název komponenty. Ovšem názvy delší než 12 znaků už nejsou jednořádkové a je proto nutné popisový blok komponenty překreslit na větší. Z těchto důvodů bylo vytvořeno rozhraní *TextModification*, které definuje metody pro změnu textu a pro získání textu z popisového bloku komponenty. Toto rozhraní implementují všechny třídy popsané v podkapitole 4.4.1 Grafické znázornění komponenty. Ukázku implementace lze nalézt v příloze B.2 Ukázka metody *textModify(String text)* ve třídě *PrimEventPane*.

Propojení komponent čárami mezi sebou

Jednotlivé značky FTA v diagramu jsou mezi se propojeny spojovacími čárami (viz. kapitola 2.4 Obecný popis FTA). Na rozdíl od papíru v aplikaci lze jednotlivé komponenty posunovat po pracovní ploše. Aby byl zachován stejný význam diagramu při posunutí komponenty (spojovací čára bude stále spojovat posunutou komponentu s jejím předkem), bylo zapotřebí při změně souřadnic komponenty přepočítat i souřadnice začátku nebo konce spojovací čáry.

V JavaFX knihovně je třída *Line*, pomocí které lze vykreslit čáru. Pro vytvoření čáry v JavaFX postačí vytvořit instanci třídy *Line* a zadat jí souřadnice začátku a konce čáry. Tyto souřadnice jsou v této třídě reprezentovány pomocí JavaFX obalovací třídy *DoubleProperty*. Výhodou této obalovací třídy je, že umožňuje propojení s jinou třídou tohoto typu tak, že změna hodnoty v jedné z nich se ihned

projeví v té druhé. Díky této funkci třídy *DoubleProperty* bylo vytvořeno rozhraní *PossitionConnector*, které implementují třídy popsané v podkapitole 4.4.1 Grafické znázornění komponenty. Toto rozhraní definuje čtyři metody (viz kód 4.1 Rozhraní *PossitionConnector*):

- getTopX()
- getTopY()
- getBottomX()
- getBottomY()

Kód 4.1: Rozhraní *PossitionConnector*

```
public interface PositionConnector {

   //Vraci x-ovou souradnici bodu v pulce horni hrany
      komponenty
   public DoubleProperty getTopX();
   //Vraci y-ovou souradnici bodu v pulce horni hrany
      komponenty
   public DoubleProperty getTopY();
   //Vraci x-ovou souradnici bodu v pulce spodni hrany
      komponenty
   public DoubleProperty getBottomX();
   //Vraci y-ovou souradnici bodu v pulce spodni hrany
      komponenty
   public DoubleProperty getBottomY();

}
```

4.4.2 Ukládání a načítání

Uložení projektu v aplikaci znamená uložit jak strom poruchových stavů, tak i jeho vizuální reprezentaci. Z počátku byla snaha ukládat data do XML dokumentu (viz kód 4.2 Návrh XML souboru). Jednotlivé komponenty měly být reprezentovány stejnojmennými značkami. Každá tato značka měla obsahovat značku vyjadřující pozici komponenty na pracovní ploše v aplikaci, pravděpodobnost poruchy na dané komponentě a potomky, které jsou k této komponentě připojeny.

Během návrhu podoby XML dokumentu se nakonec došlo k závěru, že tento způsob ukládání dat není dostatečně univerzální a dostatečně popisující. Pokud by došlo v programu ke změně struktury například třídy implementující rozhraní *IFTANode* muselo by dojít nejen k úpravě podoby XML dokumentu, ale i k úpravě

algoritmu, který by měl za úkol transformovat projekt do tohoto XML dokumentu. Dále by bylo složité znovu rekonstruovat správně strom poruch, který by obsahoval duplicitní základní události.

Kód 4.2: Návrh XML souboru

```
<fta>
   <root>
      <position>
         <x><\x>
         <y><\y>
      <\position>
      <probability><\probability>
      <children>
         <and>
            <position>
               <x><\x>
               <y><\y>
            <\position>
            <probability><\probability>
            <children>
            ...
            <\children>
         <\and>
      <\children>
   </root>
</fta>
```

Přešlo se tedy ke způsobu ukládání dat do binárního souboru[1]. Využilo se vlastnosti jazyka Java, který dokáže všechny své objekty, které implementují rozhraní *Serializable*[2], uložit do souboru a ze souboru tyto objekty zase načíst. Rozhraní *IFTANode* implementuje *Serializable* a tím pádem je serializovatelné a i všechny třídy, které toto rozhraní *IFTANode* implementují. Knihovna JavaFX ovšem nemá serializovatelné třídy, proto byla vytvořena pomocná třída *SerializablePane*, která je serializovatelná a nese informace o pozici grafické komponenty, jaký typ komponenty je a s kým byla grafická komponenta případně spojená čárou (viz kód 4.3 Třída *SerializablePane*). Dále se jen vytvořit list komponent, každá vyjádřena třídou *SerializablePane*, tento list je předán metodě *save(...)* (viz příloha B.3 Metoda pro ukládání strom poruch do souboru ve třídě *ObjectFileOperator*) ve třídě *ObjectFileOperator*, která se nachází v balíčku *utils.manipulator*.

[1] Reprezentace dat do binárního souboru znamená, že tento soubor není pro lidi čitelný.
[2] Rozhraní *Serializable* je součástí jazyka Java.

Kód 4.3: Třída *SerializablePane*

```
public class SerializablePane implements Serializable {

   //uzel reprezentujici jednu komponentu FTA
   private IFTANode node;
   //typ komponenty, ktera je reprezentovana uzlem node
   private TreePanes type;
   //pozice komponenty na pracovni plose
   private double posX, posY;
   //uzel, kteremu je tato komponenta pripojena
   private SerializablePane lineTo;

   //Konstruktor
   public SerializablePane(IFTANode node, double posX,
      double posY, TreePanes type) {
       this.node = node;
       this.posX = posX;
       this.posY = posY;
       this.type = type;
       this.lineTo = null;
   }
   ...
}
```

Načítání ze souboru funguje v podstatě inverzním způsobem k ukládání. Program si nejprve ze souboru načte všechny objekty typu *SerializablePane* a dále jeden objekt po druhém přečte a sestaví vizuálně podle nich celý strom poruchových stavů.

4.4.3 Majoritní hradlo (Hradlo M/N)

Na tomto hradle nastane porucha jen tehdy pokud vzniknou poruchy na m potomků tohoto hradla z celkových n. Pokud je toto hradlo použito v aplikaci, lze na takovémto stromě vykonávat pouze kvantitativní analýzu, protože pro kvalitativní analýzu se pak musí používat mnohem komplexnější metody než jsou použity v této aplikaci[3]. Výpočet pravděpodobnos-ti výskytu události na tomto hradle se v aplikaci počítá pomocí vzorců (4.1), kde

- m je minimální počet potomků, na kterých nastala porucha
- n je počet potomků hradla
- P_i je pravděpodobnost výskytu i-tého potomka hradla

[3]Pro hradlo 3 z 5, kde potomci hradla jsou A, B, C, D, E, by všechny minimální kritický řezy byly: $ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE, CDE$.

- P_{prum} je průměrná pravděpodobnost všech potomků hradla
- P_{hradlo} je pravděpodobnost výskytu události na hradle

$$\begin{aligned} P_{prum} &= \frac{\sum_{i=1}^{n} P_i}{n} \\ P_{hradlo} &= \sum_{i=m}^{n} \binom{n}{i} \cdot P_{prum}^{i} \cdot (1 - P_{prum})^{n-i} \end{aligned} \tag{4.1}$$

Tento způsob výpočtu pravděpodobnosti výskytu události na tomto hradle není úplně přesný. Nepřesnost výpočtu touto metodou bude ukázáno dále.

Nepřesnost výpočtu na majoritním hradle

Při výpočtu pravděpodobnosti události na tomto hradle vzniká problém, pokud se vypočítává z neshodných pravděpodobností jeho přímých potomků. Proto byl zvolen jednodušší příklad, kdy pravděpodobnosti těchto potomků byly zprůměrovány. Proto je nutné stanovit chybu tohoto zjednodušení. Výpočet chyby je proveden na nejčastějším systému dva ze tří.

Jsou definovány tři události s konstantními pravděpodobnostmi P_1, P_2 a P_3, kde $P_1 < P_2 < P_3$. Nechť tyto tři události jsou potomkem majoritního hradla. Výskyt poruchy na tomto hradle nastane tehdy pokud nastala porucha alespoň na dvou ze tří jeho potomků. To znamená, že se jedná o hradlo dva ze tří.

Při výpočtu pravděpodobnosti na majoritním hradle se používá vzorce (4.1). Skutečná hodnota pravděpodobnosti na majoritním hradle dva ze tří se vypočítá následovně:

$$\begin{aligned} P_{hradlo} = {} & P_1 \cdot P_2 \cdot P_3 + P_1 \cdot P_2 \cdot (1 - P_3) \\ & + P_1 \cdot (1 - P_2) \cdot P_3 + (1 - P_1) \cdot P_2 \cdot P_3 \end{aligned} \tag{4.2}$$

Nechť pravděpodobnost P_2 je rovna průměru hodnot všech pravděpodobností potomků na hradle:

$$P_2 = R = \frac{P_1 + P_2 + P_3}{3} \tag{4.3}$$

Dále lze vyjádřit P_1 jako $P_1 = P_2 - \Delta$ a P_3 jako $P_3 = P_2 + \Delta$, kde $\Delta = P_2 - P_1 \cap \Delta = P_3 - P_2$. Potom musí platit:

$$\begin{aligned} R^3 + 3 \cdot R^2 \cdot (1 - R) \approx {} & (R - \Delta) \cdot R \cdot (R + \Delta) \\ & + (R - \Delta) \cdot R \cdot (1 - (R + \Delta)) \\ & + (R - \Delta) \cdot (1 - R) \cdot (R + \Delta) \\ & + (1 - (R - \Delta)) \cdot R \cdot (R + \Delta) \end{aligned} \tag{4.4}$$

Nyní několik úprav pravé strany rovnice:

$$\begin{aligned} R^3 + 3 \cdot R^2 \cdot (1 - R) \approx\ & R \cdot (R^2 - \Delta^2) + (R^2 - R\Delta) \cdot (1 - R - \Delta) \\ & + (1 - R) \cdot (R^2 - \Delta^2) \\ & + (1 - R + \Delta) \cdot (R^2 + R\Delta) \end{aligned} \tag{4.5}$$

$$\begin{aligned} R^3 + 3 \cdot R^2 \cdot (1 - R) \approx\ & R^3 - R\Delta^2 + R^2 - R\Delta \\ & - R^3 + R^2\Delta - R^2\Delta + R\Delta^2 \\ & + R^2 - \Delta^2 - R^3 + R\Delta^2 \\ & + R^2 + R\Delta - R^3 - R^2\Delta + R^2\Delta + R\Delta^2 \end{aligned} \tag{4.6}$$

$$R^3 + 3 \cdot R^2 \cdot (1 - R) \approx -2 \cdot R^3 + 2 \cdot R \cdot \Delta^2 + 3 \cdot R^2 - \Delta^2 \tag{4.7}$$

Nyní se provedou konečné úpravy levé a pravé strany rovnice:

$$3 \cdot R^2 - 2 \cdot R^3 \approx 3 \cdot R^2 - 2 \cdot R^3 + (2 \cdot R - 1) \cdot \Delta^2 \tag{4.8}$$

V rovnici (4.8) se pravá strana od levé strany liší o $(2 \cdot R - 1) \cdot \Delta^2$. To znamená, že výpočet pravděpodobnosti na majoritním hradle je s chybou druhého řádu. Ale vzhledem k tom, že Δ je rozdíl reálné hodnoty pravděpodobnosti od průměrné hodnoty (hodnoty Δ se pohybují pouze v desetinných číslech), je chybovost výpočtu pravděpodobnosti na hradle v aplikaci zanedbatelná.

4.4.4 Kvalitativní vyhodnocení FTA

Pro kvalitativní vyhodnocení stromu poruchových stavů se nejprve muselo všem základním událostem přiřadit písmeno z abecedy, aby je bylo možno prezentovat jako proměnné v booleovské funkci. Pro hledání základních událostí byl strom prohledáván do šířky[4]. Tím se zajistilo, že písmena se budou přiřazovat těmto událostem od nejvýše položených ve stromě až po nejníže položené. Ukázku kódu hledání základních událostí pomocí prohledávání stromu do šířky je v příloze B.1 Vyhledání základních událostí pomocí hledání do šířky.

Booleovská funkce, která reprezentuje strom poruchových stavů, byla získána pomocí průchodu stromu do hloubky. Metoda *getBoolExpression()*, která je definována v rozhraní *IFTANode*, je implementována ve všech třídách reprezentující komponenty stromu poruch. Metoda je rekurzivní a vrací booleovskou funkci vyjadřující strom poruch.

[4]Prohledávání do šířky je grafový algoritmus pro postupné procházení stromu po patrech.

Úprava booleovské funkce

Booleovská funkce obdržená od metody *getBoolExpression()* je potřeba převést na disjunktní normální formu. Pro tuto potřebu byla napsána třída *BracketsSolver*, která dokáže roznásobit závorky vyskytující se v booleovské funkci.

Pro použití metody na minimalizaci booleovské funkce je potřeba funkci přepsat na tvar součtu mintermů. Pro tyto účely byla vytvořena třída *TruthTable*, která dokáže přepsat jakoukoliv booleovskou funkci v disjunktní normální formě na pravdivostní tabulku. Součet mintermů se zapisuje jako čísla řádků pravdivostní tabulky, ve kterých tabulka nabývá hodnoty '1'.

Minimalizace booleovské funkce

K dokončení nalezení množiny všech minimálních kritických řezů je potřeba minimalizovat booleovskou funkci. Booleovská funkce se minimalizuje pomocí booleovské redukce, Karnaughových map nebo pomocí Quine-McCluskeyho algoritmu. Pro minimalizaci funkce je použita metoda Quine-McCluskey, protože se tato metoda snadněji implementuje na strojích než metoda pomocí Karnaughových map.

Již naimplementovanou Quine-McCluskeyho algoritmus v jazyce Java byl nalezen na internetu z důvodu náročnosti implementace. Autorem tohoto kódu je pan Ahmed Yakout, který poskytl kód volně šiřitelný pod licencí MIT. Kód algoritmu byl nalezen na stránce `https://github.com/yakout/quine-mccluskey`. Tomuto programu byla předána množina mintermů a zpátky vrácena je minimalizovaná booleovská funkce. Podle této funkce bylo pak možné zobrazit základní události, které patří do množiny minimálních kritických řezů.

4.4.5 Kvantitativní vyhodnocení FTA

U kvantitativního hodnocení stromu poruch se muselo nejprve vyřešit jestli se ve stromě poruchových stavů vyskytují duplicitní základní události. Pokud ve stromě neexistovaly duplicity, mohlo být použita přímá metoda pro vyhodnocení stromu poruch. Tato metoda pro vyhodnocení stromu je definovaná už v rozhraní *IFTANode* a její implementace se liší podle komponenty, ve které je implementována. Pokud komponenta dědila ze třídy *AEvent*, metoda vracela přímo hodnotu pravděpodobnosti výskytu události nadefinovanou uživatelem. Jestliže komponenta byla potomkem třídy *AGate*, metoda byla implementována podle vzorce pro dané hradlo z kapitoly 2.6.3 Kvantitativní analýza FTA.

Ovšem pokud se vyskytovaly duplicitní základní události ve stromě poruchových stavů musí být použita metoda výpočtu pomocí minimálních kritických řezů. Nejprve byl vyhodnocen strom poruchových stavů kvalitativně. Získaná množina minimálních kritický řezů byla poté transformována zpět do booleovské funkce.

Binarní rozhodovací diagram

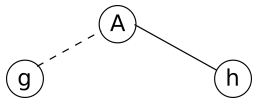
Funkce byla dále převedena do disjunktní formy za pomoci binárního rozhodovacího diagramu. Binární rozhodovací diagram je datová struktura, která se používá pro reprezentaci booleovské funkce v informatice. Jedná se o datovou strukturu binární strom, kde jednotlivé uzly reprezentují proměnné booleovské funkce a listy tohoto stromu nabývají logických hodnot. Levý potomek uzlu reprezentuje funkci, kde za proměnnou reprezentovanou tímto uzlem lze dosadit ve funkci logickou '0'. Pravý potomek uzlu zas reprezentuje funkci, kde proměnná ve funkci nabývá hodnoty logická '1'. Listy tohoto stromu jsou uzle, které representují logickou '1' nebo '0'.

Pro nalezení disjunktní formy funkce je třeba binární rozhodovací diagram převést nejprve na seřazený binární rozhodovací diagram a poté na redukovaný seřazený binární rozhodovací diagram. Seřazený binární rozhodovací diagram je takový diagram, kde uzly mezi sebou přechází v pořadí, které určuje seřazený list proměnných vyskytujících se v dané funkci. To znamená, že pokud by v listu šly po sobě proměnný A, B, C, tak v diagramu nesmí nastat, že uzel reprezentující proměnnou B bude mít mezi potomky uzel reprezentující proměnnou A.

Nechť je definována booleovská funkce $f = ac+bc$. Pro sestavení binárního rozhodovacího diagramu z této funkce se použije tzv. dekompozice booleovské funkce. Dekompozice funkce spočívá v přepsání funkce na součet dvou nových funkcí, u kterých je vyjádřena jedna z proměnných. Příklad (4.9) znázorňuje dekompozici funkce $f = ac + bc$ pro proměnnou 'a'.

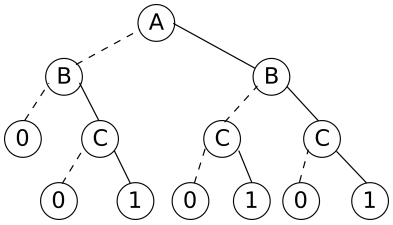
$$\begin{aligned} f &= ac + bc \\ f_{\overline{a}} &= f_{(a=0)} = bc \\ f_a &= f_{(a=1)} = c + bc \\ f &= a \cdot f_a + \overline{a} \cdot f_{\overline{a}} \end{aligned} \tag{4.9}$$

Jestliže $g = bc$ a $h = c + bc$ potom binární rozhodovací diagram funkce $f = ac + bc$ vypadá jako na obrázku 4.1, kde levý potomek uzlu A je roven funkci g a pravý potomek je roven funkci h.



Obrázek 4.1: Část binárního rozhodovacího diagramu

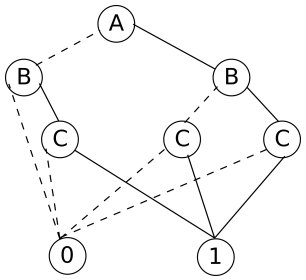
Pokud se pokračuje v dekompozici dále, vznikne binární rozhodovací diagram zobrazený na obrázku 4.2.



Obrázek 4.2: Seřazený binárního rozhodovací diagram funkce $f = ac + bc$

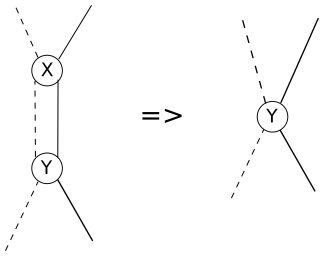
Redukovaný seřazený binární rozhodovací diagram vznikne vykonáním úprav nad seřazeným binárním rozhodovacím diagramem v předepsaném pořadí:

1) Sjednotit listy reprezentující stejné hodnoty do jednoho listu (viz obrázek 4.3).



Obrázek 4.3: Sjednocení stejných listů diagramu

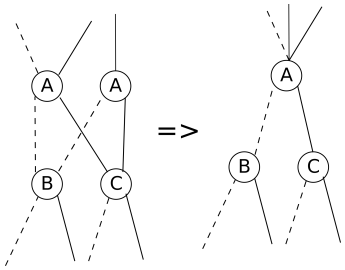
2) Pokud levý potomek uzlu je stejný jako pravý potomek tohoto uzlu, nahradí se tento uzel jeho potomkem (viz obrázek 4.4).



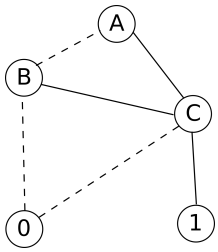
Obrázek 4.4: Nahrazení uzlu BDD

3) Existují-li dva uzle reprezentující stejnou proměnnou funkce, jejichž levý potomek je jeden a ten samý a pravý potomek taktéž, sjednotíme tyto uzly do jednoho (viz obrázek 4.5).

Postup 2) a 3) se opakuje tak dlouho až už nelze uplatnit na diagramu ani jedna úprava a výsledný diagram je redukovaný seřazený binární rozhodovací diagram. Výsledný rozhodovací diagram po redukci je vyobrazen na obrázku 4.6.[15]

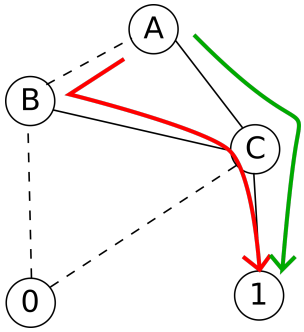


Obrázek 4.5: Sjednocení duplicitních uzlů BDD



Obrázek 4.6: Redukovaný BDD funkce $f = ac + bc$

Binární rozhodovací diagram byl podle výše uvedeného popisu naprogramován do třídy *BddTree*. Po předání booleovské funkce si třída automaticky vytvoří seřazený binární rozhodovací diagram. Dále stačí na strom zavolat metodu *minimizedBdd()*, která transformuje diagram na redukovaný. Pro získání disjunktní formy booleovské funkce byl prohledán strom do hloubky a byly hledány všechny cesty, které končili v listě s logickou hodnotou '1'. Disjunktní forma booleovské funkce z výše uvedeného příkladu má předpis $f = \overline{a}bc + ac$, kde k získání sčítance $\overline{a}bc$ znázorňuje červená šipka a k získání sčítance ac znázorňuje zelená šipka na obrázku 4.7. Každá cesta



Obrázek 4.7: Postup pro získání disjunktní formy booleovské funkce z BDD

byla přepsána na součin proměnných, přes které cesta vedla. Pokud v určitém uzlu při průchodu cestou se muselo jít dále přes jeho levého potomka, vyjadřuje se tento uzel ve funkci jako negace proměnné, kterou reprezentuje.

Výsledná funkce byla poté sestavena jako součet všech nalezených cest. Dále byly dosazeny za proměnné pravděpodobnost výskytu událostí, které tyto proměnné reprezentují. Jestliže se objevila ve funkci negovaná proměnná stačilo hodnotu pravděpodobnosti odečíst od 1.

Analýza z CSV souboru

Kvantitativní analýzu lze provést i na strom popsaný v CSV souboru (viz kapitola 6.1 Struktura CSV souboru). Strom popsaný v CSV souboru je však omezený jen na komponenty:

- Vrcholová událost
- Primární událost
- Hradlo AND
- Hradlo OR
- Hradlo M z N (Majoritní hradlo)

Tento CSV soubor aplikace načte, sestaví podle něj strom poruch a kvantitativně tento strom vyhodnotí. Aplikace neumí podle tohoto souboru strom zobrazit.

5 Testování aplikace

Vytvořená aplikace byla otestována několika vytvořenými stromy poruch. Testování bylo uskutečněno tak, že se kvalitativní i kvantitativní analýza uskutečnila jak pomocí aplikace, tak i ručními výpočty.

5.1 První testovací strom poruch

První testovací strom poruch je vyobrazen na obrázku A.3.1 v příloze A.3. Pravděpodobnost výskytu poruchy jednotlivých základních událostí A, B a C jsou:

- $P(A) = 0.3$
- $P(B) = 0.2$
- $P(C) = 0.1$

5.1.1 Kvalitativní analýza

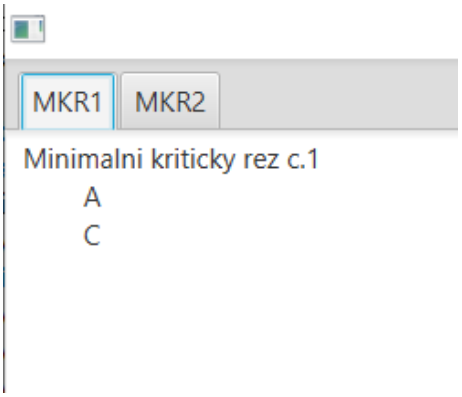
Přepis stromu do booleovské funkce vypadá následovně:

$$\begin{aligned} f &= a \cdot (b + c) \\ f &= ab + ac \end{aligned} \tag{5.1}$$

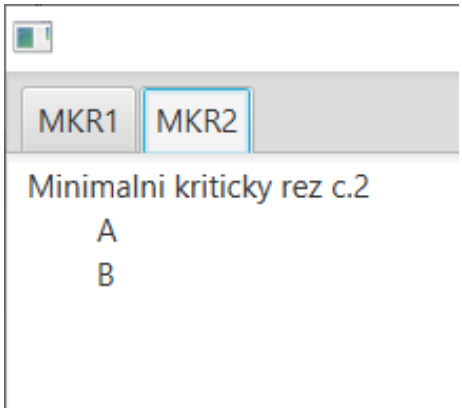
Minimální kritické řezy při ručním výpočtu vyšly:

- Minimální kritický řez č. 1 – ab
- Minimální kritický řez č. 2 – ac

Výsledek, ke kterému došla aplikace je vyobrazena na obrázcích 5.1 a 5.2). Výsledky, které vypočítala aplikace, se shodují s výsledky, ke kterým se došlo pomocí výpočtů ručně.



Obrázek 5.1: Minimální kritický řez č. 1



Obrázek 5.2: Minimální kritický řez č. 2

5.1.2 Kvantitativní analýza

Nejprve se musí vypočítat pravděpodobnost poruchy na hradle $G2$ pomocí vzorce (2.11) z kapitoly 2.6.3 Kvantitativní analýza FTA:

$$\begin{aligned} G2 &= 1 - [(1 - P_B) \cdot (1 - P_C)] \\ G2 &= 1 - [(1 - 0.2) \cdot (1 - 0.1)] \\ G2 &= 0.28 \end{aligned} \tag{5.2}$$

Dále lze vypočítat pravděpodobnost poruchy na hradle $G1$ a tím se vypočítá i pravděpodobnost výskytu vrcholové události. Pro výpočet na hradle $G1$ se použije vzorec (2.10) z kapitoly 2.6.3 Kvantitativní analýza FTA:

$$\begin{aligned} G1 &= G2 \cdot P_A \\ G1 &= 0.28 \cdot 0.3 \\ G1 &= 0.084 \end{aligned} \tag{5.3}$$

Výsledek výpočtu, ke kterému došla aplikace, je zobrazen na obrázku 5.3. Výsledek vypočítaný aplikací je roven výsledku vypočítaný ručně.

Obrázek 5.3: Výsledek kvantitativní analýzy pomocí aplikace

5.2 Druhý testovací strom poruch

Druhý testovací strom poruch je zobrazen na obrázku A.3.2 v příloze A.3. Pravděpodobnost výskytu poruchy jednotlivých základních událostí A, B, C, D, E, F, H a I jsou: $P_A = 0.8, P_B = 0.7, P_C = 0.6, P_D = 0.5, P_E = 0.4, P_F = 0.1, P_H = 0.2, P_I = 0.3$.

5.2.1 Kvalitativní analýza

Přepis stromu do booleovské funkce:

$$\begin{aligned} f &= a \cdot G2 \cdot b \\ f &= a \cdot (G3 + G4) \cdot b \\ f &= a \cdot (c \cdot G5 + (d + G6)) \cdot b \\ f &= a \cdot (c \cdot (e + f) + (d + h \cdot i)) \cdot b \end{aligned} \tag{5.4}$$

Dále se provede několik úprav nad touto funkcí:

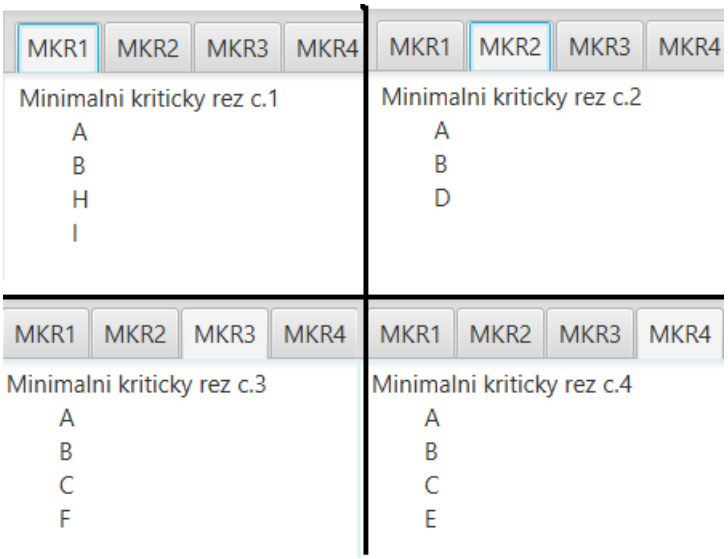
$$\begin{aligned} f &= a \cdot (c \cdot (e + f) + (d + h \cdot i)) \cdot b \\ f &= a \cdot (c \cdot e + c \cdot f + d + h \cdot i) \cdot b \\ f &= a \cdot b \cdot c \cdot e + a \cdot b \cdot c \cdot f + a \cdot b \cdot d + a \cdot b \cdot h \cdot i \end{aligned} \tag{5.5}$$

Pomocí ručního výpočtu vyšly tyto minimální kritické řezy:

- Minimální kritický řez č. 1 – $ABCE$
- Minimální kritický řez č. 2 – $ABCF$

- Minimální kritický řez č. 3 – ABD
- Minimální kritický řez č. 4 – $ABHI$

Aplikace kvalitativní analýzu vyhodnotila stejně jako byla vyhodnocena pomocí ručního výpočtu (viz obrázek 5.4).



Obrázek 5.4: Výsledek kvalitativní analýzy pomocí aplikace pro druhý test. strom

5.2.2 Kvantitativní analýza

Nejprve se vypočítá pravděpodobnosti hradel $G5$ a $G6$. Poté se mohou dopočítat pravděpodobnosti hradel $G3$ a $G4$. Nakonec se vypočte pravděpodobnost $G2$ a následně $G1$. Při vypočítání pravděpodobnosti na $G1$ získáme pravděpodobnost výskytu vrcholové události. Níže je ukázán postup výpočtu.

$$\begin{aligned} G5 &= 1 - ((1 - P_E) \cdot (1 - P_F)) \\ G5 &= 1 - (0.6 \cdot 0.9) \\ G5 &= 0.46 \end{aligned} \tag{5.6}$$

$$\begin{aligned} G6 &= P_H \cdot P_I \\ G6 &= 0.2 \cdot 0.3 \\ G6 &= 0.06 \end{aligned}$$

Dále je výpočet $G3$ a $G4$:

$$\begin{aligned} G3 &= G5 \cdot P_C \\ G3 &= 0.46 \cdot 0.6 \\ G3 &= 0.276 \end{aligned} \tag{5.7}$$

$$\begin{aligned} G4 &= 1 - ((1 - P_D) \cdot (1 - G6)) \\ G4 &= 1 - (0.5 \cdot 0.94) \\ G4 &= 0.53 \end{aligned}$$

Nyní se vypočte pravděpodobnost na hradle $G2$:

$$\begin{aligned} G2 &= 1 - ((1 - G3) \cdot (1 - G4)) \\ G2 &= 1 - (0.724 \cdot 0.47) \\ G2 &= 0.65972 \end{aligned} \tag{5.8}$$

Nakonec se vypočítá pravděpodobnost hradla $G1$ a tím pádem i pravděpodobnost výskytu vrcholové události:

$$\begin{aligned} G1 &= P_A \cdot G2 \cdot P_B \\ G1 &= 0.8 \cdot 0.65972 \cdot 0.7 \\ G1 &= 0.3694432 \end{aligned} \tag{5.9}$$

Pravděpodobnost výskytu vrcholové události vyšla při ručním počítání 0.3694432. Aplikace vypočítala tuto pravděpodobnost na hodnotu 0.369. Vzhledem k tomu, že aplikace počítá na tři platná desetinná místa, po zaokrouhlení výsledku z ručního výpočtu na tři desetinná místa je 0.369.

6 Popis aplikace

V této kapitole bude popsána aplikace z pohledu uživatele. Bude zde popsáno grafické rozhraní aplikace a jak se s aplikací pracuje.

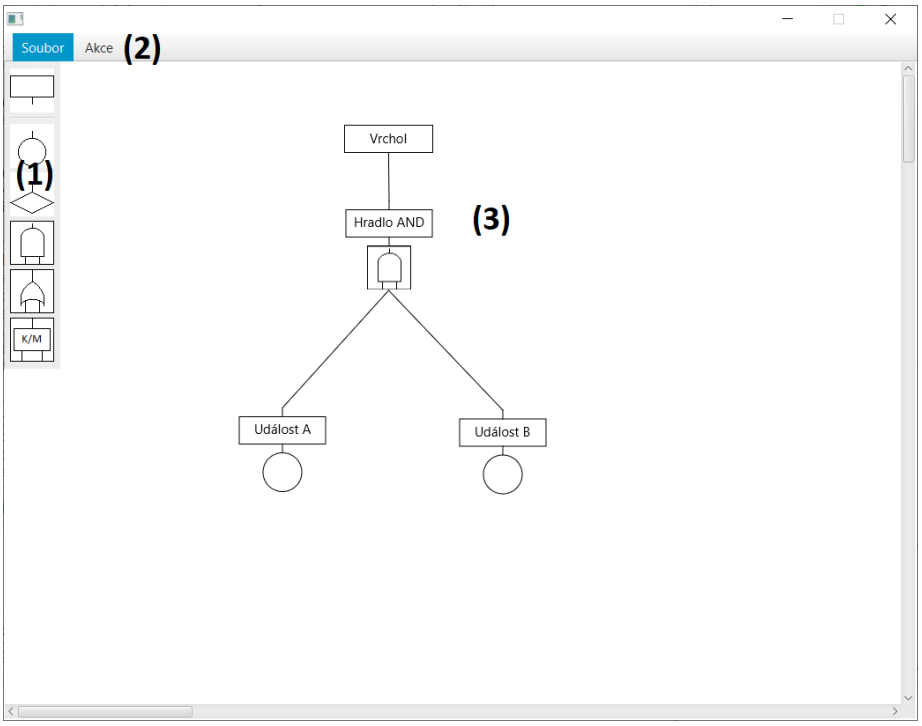
6.1 Hlavní okno programu

Úvodní okno, které se zobrazí při spuštění program. Toto okno aplikace je vyobrazeno na obrázku 6.1. Skládá se ze tří částí:

(1) Panel nástrojů

(2) Menu

(3) Pracovní plocha



Obrázek 6.1: Hlavní okno aplikace

Panel nástrojů

Na panelu nástrojů se nalézají tlačítka, které reprezentují jednotlivé komponenty stromu poruch (viz 2.5 Používané značky v FTA). Zatím jsou v rámci aplikace naimplementovány komponenty:

- Vrcholová událost
- Základní událost
- Nerozvíjená událost
- Hradlo AND
- Hradlo OR
- Majoritní hradlo (Hradlo M/N)

Pracovní plocha

Na pracovní ploše se zobrazuje a sestavuje strom poruch tak, že se z panelu nástrojů přetažením vytáhne požadovaná komponenta na plochu. Kliknutím pravým tlačítkem myši na komponentu vyskočí kontextové menu (viz obrázek C.1). Toto menu obsahuje položky:

- **Připojit** – po vybrání této položky a kliknutí na jinou komponentu v pracovní ploše se vytvoření propojení mezi nimi.
- **Upravit** – vyvolá nové okno pro úpravu komponenty (viz kapitola 6.2).
- **Duplikovat** – vytvoří kopii komponenty tak, že změna v jedné z nich se projeví i u druhé.
- **Zkopírovat** – vytvoří novou komponentu, která má nastavené stejné hodnoty a parametry jako její předloha.
- **Smazat** – smaže komponentu z pracovní plochy i se všemi propojeními.

Menu

Menu obsahuje dvě položky: **Soubor**, **Akce**. Po kliknutí na položku **Soubor** se rozvine menu, ve kterém se nachází tlačítka:

- **Nový** – vytvoří nový projekt.
- **Uložit** – uloží projekt do vybrané složky.
- **Nahrát** – nahraje projekt z vybraného souboru.

Při výběru **Akce** vyskočí na výběr ze dvou položek:

Obrázek 6.2: Kontextové menu komponenty

- **Spustit kvalitativní výpočet** – spustí kvalitativní analýzu vytvořeného stromu.
- **Spustit kvantitativní výpočet** – spustí kvantitativní analýzu vytvořeného stromu.
- **Spustit kvantitativni výpočet z CSV** – spustí kvantitativní analýzu na strom popsaný v csv souboru (struktura CSV souboru je popsána v kapitole 6.1).

Po skončení každé z analýz vyskočí nové okno s výsledky (viz kapitola 6.3).

Struktura CSV souboru

CSV soubor má následující strukturu:

Kód 6.1: Struktura CSV souboru

```
FTA
1;TOP;nazev;potomek uzlu vyjadreny cislem v souboru
cislo komponenty;(OR/AND);nazev;potomci vyjadreny
   cislem uzlu
nebo
cislo komponenty;PRIM;nazev;(C/L);hodnota vyjadrena
   vedeckym zapisem cisla
```

Soubor musí vždy mít na prvním řádku slovo ***FTA***. Jednotlivé řádky souboru reprezentují vždy jednu komponentu stromu. Řádky reprezentující hradla stromu mají tvar:

- ***číslo komponenty;OR/AND/MN;název komponenty;potomci vyjádřený jejich číslem v CSV souboru(;minimální počet poruchových komponent (pouze pro MN hradlo))***

Řádky reprezentující základní události mají obecný tvar:

- ***číslo komponenty;PRIM;název události;C/L;hodnota vyjádřena vědeckým zápisem čísla***

Význam a popis jednotlivých položek v CSV souboru:

- **číslo komponenty** – číslo komponenty v rámci CSV souboru, čísla musí jít popořadě.
- **OR/AND/MN/PRIM/TOP** – zkratky typu komponenty, může být použita vždy jen jedna na komponentu. PRIM – primární událost, TOP – vrcholová událost.
- **název** – název události ve stromě poruch
- **C/L** – pouze pro typ komponenty PRIM, C – následující hodnota je konstatní pravděpodobnost, L – následující hodnota je intenzita poruchy.
- **hodnota** – pouze u komponentů typu PRIM, hodnota zapsaná vědeckým zápisem čísla.
- **potomek/ci komponenty** – číslo nebo čísla komponent oddělená čárkou v rámci CSV souboru
- **minimální počet poruchových komponent** – pouze pro hradlo MN, určuje minimální počet poruchových komponent, aby nastala událost na tomto hradle.

Kód 6.2: Příklad CSV souboru

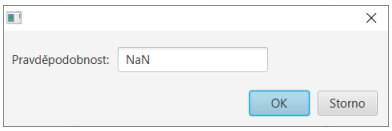
```
FTA
1;TOP;Top;2
2;AND;AND hradlo;3,4,5
3;PRIM;A;C;0.5
4;PRIM;B;L;1e-2
5;PRIM;C;C;2e-1
```

6.2 Okno úpravy komponenty

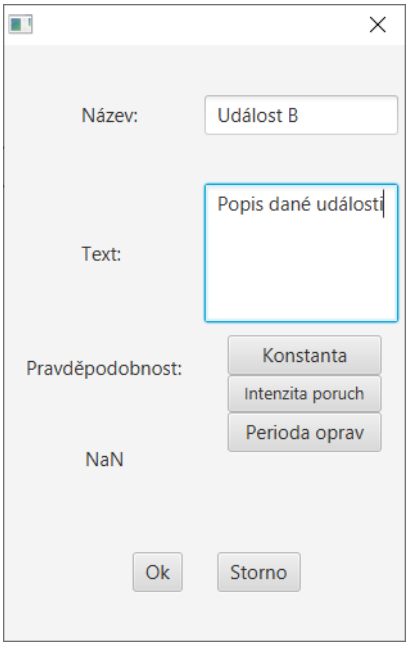
Toto okno vyskočí vždy po vybrání položky Upravit z kontextového menu dané komponenty. Toto okno obsahuje:

- **Název** – název dané komponenty.
- **Text** – detailní popis dané komponenty.

- **Pravděpodobnost** – zobrazuje s jakou pravděpodobností daná událost nastane. Číselná hodnota se zobrazí jen pokud byla události nastavena konstantní pravděpodobnost výskytu nebo po vykonání kvantitativního výpočtu.
- **Konstanta** – po stisknutí tohoto tlačítka vyskočí okno (viz obrázek 6.3) pro zadání konstantní pravděpodobnosti poruchy ve formě desetinného čísla v rozmezí od 0 do 1.
- **Intenzita poruch** – po stisknutí tohoto tlačítka vyskočí okno podobné oknu u konstanty pro zadání intenzity poruchy ve formě vědeckého zápisu čísla (např. 1.25E-2 = 0.0125).
- **Perioda oprav** – nastavuje se hodnota po jaké časové periodě se komponenta opraví.
- **M/N** – pouze u hradla M/N, nastavuje se kolik minimálně je potřeba porouchaných komponent u hradla, aby vznikla poruchová událost na hradle M/N.



Obrázek 6.3: Okno pro zadání hodnoty pravděpodobnosti



Obrázek 6.4: Okno úpravy komponenty

6.3 Okno výsledků analýzy

Okno výsledků prezentuje výsledky získané buď kvalitativní nebo kvantitativní analýzou stromu poruch. Pokud je použita kvalitativní analýza v okně se zobrazí záložky s jednotlivými minimálními řezy (viz Obr. A.2). Jestliže byla provedena kvantitativní analýza, v okně budou záložky s grafy jednotlivých primárních událostí a záložka s grafem vrcholové události (neopravitelný systém viz Obr. A.3, opravitelný systém viz Obr. A.4). Tento graf reprezentuje pravděpodobnost výskytu poruchy v závislosti na čase. Graf na vybrané záložce lze uložit do souboru i s výpisem hodnot pomocí tlačítka *Uložit*. Pro uložení všech grafů s výpisy hodnot lze využít tlačítko *Uložit vše*.

Na obrázku A.5 je graf opravitelného systému. Na grafu je vidět, že má exponenciální průběh. V grafu je také znázorněno, že některé komponenty jsou opravitelné. Oprava komponenty je vidět v lokálních minimech grafu. Ačkoliv jsou některé komponenty periodicky opravovány je graf rostoucí. Toto zapříčinily neopravitelné komponenty stromu.

6.4 Generátor náhodného stromu do CSV souboru

Byl vytvořen generátor náhodného stromu do CSV souboru (CSV soubor popsán v kapitole 6.1 Struktura CSV souboru). Při spuštění tohoto generátoru se program zeptá na:

- Z kolika hradel se strom má skládat.
- Maximální počet potomků, které může jedno hradlo mít.
- Název souboru, do kterého se vše uloží.

7 Závěr práce

Cílem této práce bylo seznámit se s analýzou pomocí metody stromu poruchových stavů a vytvořit prototyp aplikace, která tuto metodu bude implementovat. Metoda stromu poruchových stavů je velmi silný analytický nástroj systému řízení jakosti, který umožní analytikovi vyhodnotit spolehlivost systému, určit pravděpodobnost selhání systému a případně i nalézt řešení pro zvýšení spolehlivosti či snížení pravděpodobnosti selhání systému. V práci je metoda analýzy stromu poruchových stavů popsána od definování použitých výrazů a značek v metodě, přes sestrojení a grafické znázornění až po vyhodnocení této analýzy.

Dále se práce zabývala již dostupnými aplikacemi, které umožňují použít tuto analytickou metodu. Byly vypracovány rešerše na tyto aplikace (viz kapitola 3), ve kterých je zmíněno o společnostech stojících za vývojem a základní orientaci v těchto aplikací. Byla snaha vybírat nejen profesionální placené produkty, ale i bezplatné verze.

Nakonec byl v práci popsán vývoj aplikace, kde byly uplatněny získané znalosti o metodě stromu poruchových stavů a vizuální stránka konkurenčních aplikací. Dále bylo v této práci popsáno řešení některých problémů při vývoji aplikace, například výpočet pravděpodobnosti na majoritním hradle stromu (viz kapitola 4.4.3 Majoritní hradlo (Hradlo M/N)) či binární rozhodovací diagram v kvantitativní analýze (viz kapitola 4.4.5 Binarní rozhodovací diagram).

V rámci této práce se vytvořila aplikace, která umožňuje seznámení se s metodou stromu poruchových stavů. Aplikace má uživatelsky přívětivé rozhraní s intuitivním ovládáním a může být využita i pro výukové účely. Aplikace by se mohla dále rozšířit například o automatické generování náhodného stromu poruch, možnost vytváření více stromů v rámci jednoho projektu, doimplementování všech značek (viz kapitola 2.5) stromu poruch nebo vyobrazení stromu poruchu v aplikaci podle CSV souboru.

Literatura

[1] LEE, W. S., D. L. GROSH, F. A. TILLMAN a C. H. LIE. Fault Tree Analysis, Methods, and Applications – A Review. IEEE Transactions on Reliability [online]. 1985, R-34(3), 194-203 [cit. 2018-02-26]. DOI: 10.1109/TR.1985.5222114. ISSN 0018-9529. Dostupné z: http://ieeexplore.ieee.org/document/5222114/

[2] ČSN EN 61025: Analýza stromu poruchových stavů (FTA). Praha: Český normalizační institut, 2007

[3] MAHEL, Roman. Analýza stromu poruchových stavů (FTA) a analýza možných vad a jejich důsledků (FMEA) procesu pájení a vodivého lepení v elektronice [online]. České vysoké učení technické v Praze, 2016 [cit. 2018-02-26]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/64869/F3-DP-2016-Mahel-Roman-Analyza%20stromu%20poruchovych%20stavu%20%28FTA%29%20a%20analyza%20moznych%20vad%20a%20jejich%20dusledku%20%28FMEA%29%20procesu%20pajeni%20a%20vodiveho%20lepeni%20v%20elektronice.pdf?sequence=-1&isAllowed=y. Diplomová práce. ČVUT, Fakulta elektrotechnická, Katedra elektroenergetiky. Vedoucí práce Doc. Ing. Pavel Mach, CSc.

[4] HOLUB, Rudolf a Zdeněk VINTR. Spolehlivost letadlové techniky [online]. Brno: Vysoké učení technické v Brně, 2001 [cit. 2018-02-26]. Dostupné z: http://lu.fme.vutbr.cz/files/SpolehlivostLetadloveTechniky.pdf

[5] ALD Reliability Engineering Ltd. ALD Fault Tree Analyser [online]. Tel Aviv: ALD [cit. 2018-12-11]. Dostupné z: http://www.fault-tree-analysis-software.com/ald-reliability-safety-engineering

[6] Reliability, Safety and Risk Assessment Software Experts. Reliability, Safety Analysis and Risk Assessment Software from Item Software [online]. Hampshire: ITEM Software [cit. 2018-12-11]. Dostupné z: http://www.itemsoft.com/about_us.html

[7] Fault Tree Analysis (FTA). Reliability, Safety Analysis and Risk Assessment Software from Item Software [online]. Hampshire: ITEM Software [cit. 2018-12-11]. Dostupné z: http://www.itemsoft.com/fault_tree.html

[8] Free Fault Tree Analysis Software - TopEvent FTA Express. TopEvent FTA - Fault Tree Analysis Software [online]. Reliotech, 2017 [cit. 2018-12-11]. Dostupné z: https://www.fault-tree-analysis.com/free-fault-tree-analysis-software

[9] EMFTA: an Open Source Tool for Fault Tree Analysis. Https://www.sei.cmu.edu/ [online]. Pittsburgh: Carnegie Mellon University [cit. 2018-12-12]. Dostupné z: https://insights.sei.cmu.edu/sei_blog/2016/07/emfta-an-open-source-tool-for-fault-tree-analysis.html

[10] RACEK, David. Programovací jazyky Java, C Sharp a C++: důležité informace a využití. Java portál [online]. Praha: Amaio Technologies, 27. 11. 2017 [cit. 2018-11-27]. Dostupné z: http://www.java.cz/article/java-42

[11] JavaFX: Getting Started with JavaFX. Help Center [online]. Redwood City (Kalifornie): Oracle, 2014 [cit. 2018-11-27]. Dostupné z: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm

[12] JavaFX Scene Builder. Oracle [online]. Redwood City (Kalifornie): Oracle, 2014 [cit. 2018-11-27]. Dostupné z: https://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

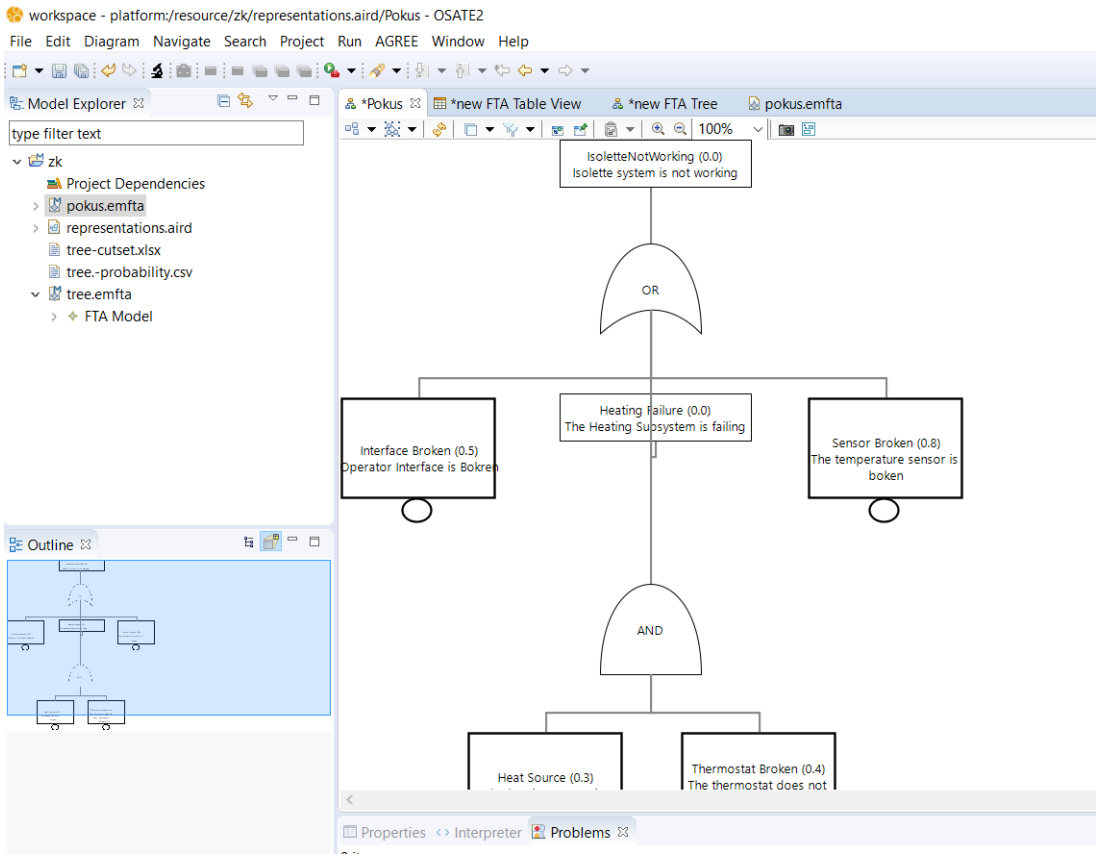
[13] IntelliJ IDEA. JetBrains [online]. Praha: JetBrains, ©2018 [cit. 2018-11-27]. Dostupné z: https://www.jetbrains.com/idea/

[14] Inkscape Overview. Inkscape [online]. Inkscape [cit. 2018-11-27]. Dostupné z: https://inkscape.org/about/

[15] PFENNING, Frank. Lecture Notes on Binary Decision Diagrams [online]. Carnegie Mellon University, 2010 [cit. 2018-11-29]. Dostupné z: https://www.cs.cmu.edu/~fp/courses/15122-f10/lectures/19-bdds.pdf

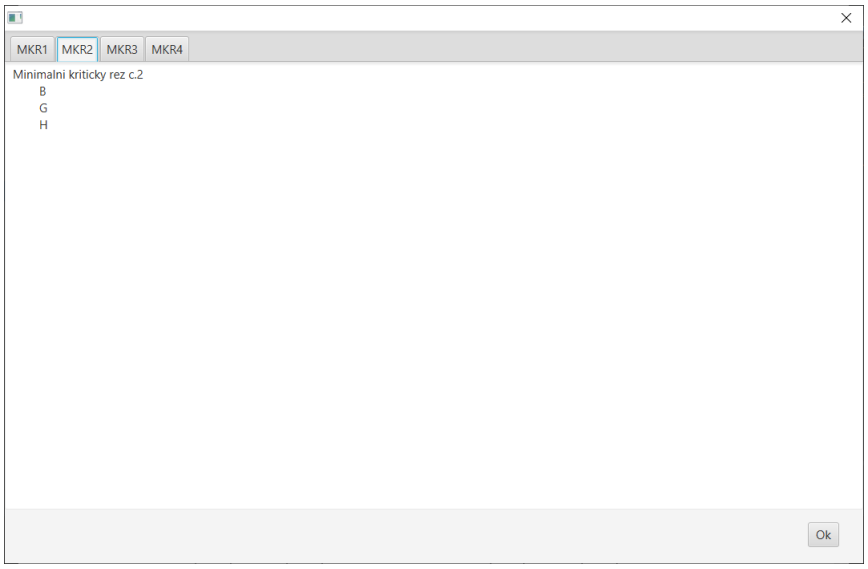
A Obrázky

A.1 Prostředí programu EMFTA

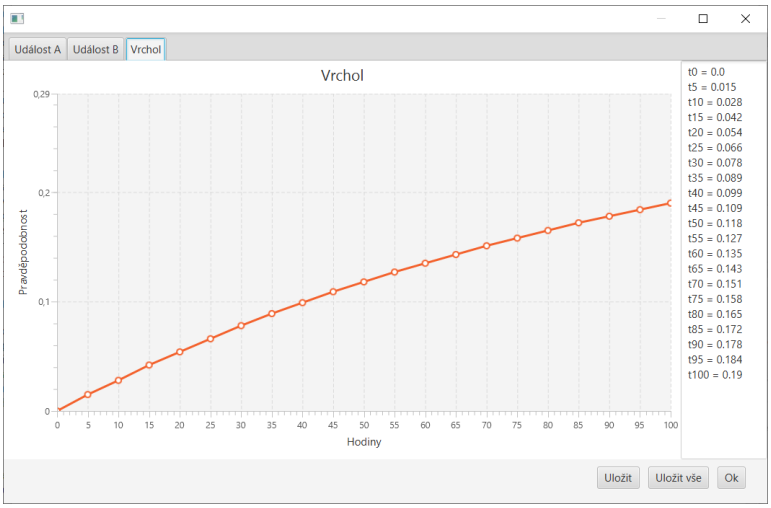


Obrázek A.1: Program EMFTA

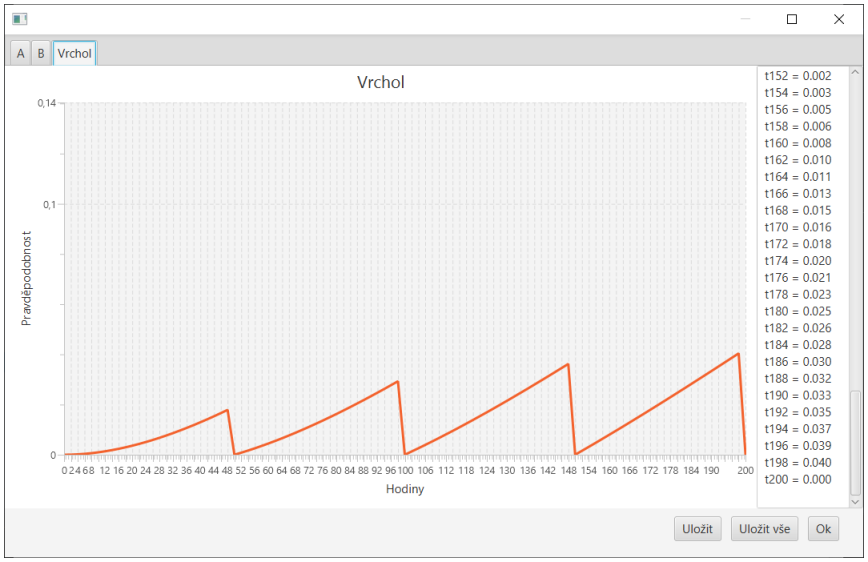
A.2 Okno výsledků



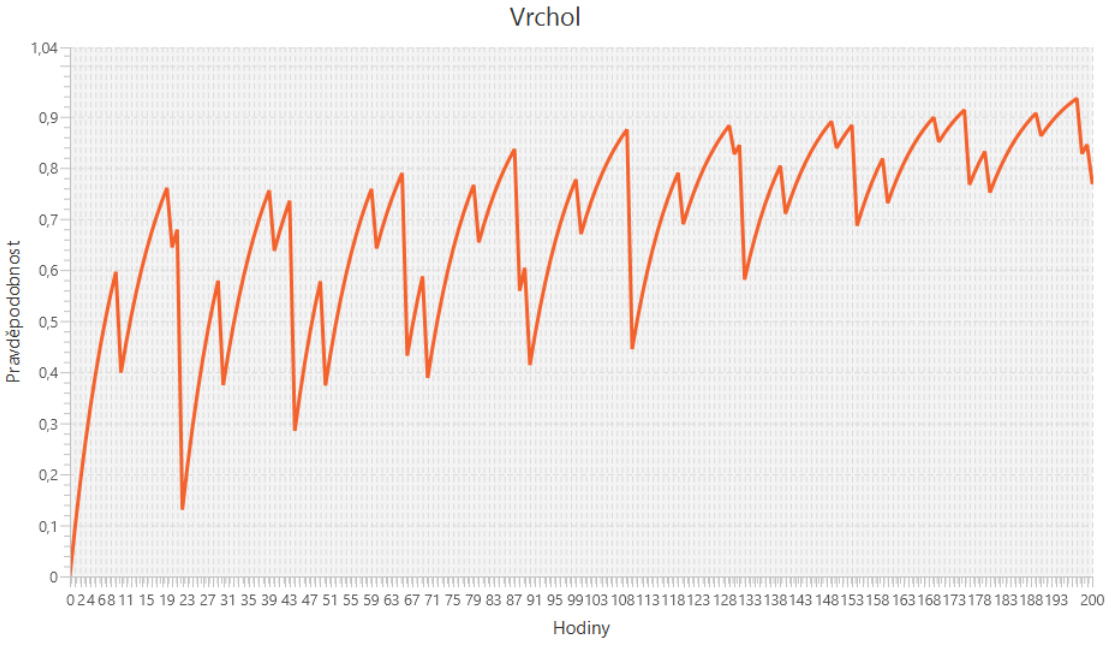
Obrázek A.2: Po kvalitativní analýze stromu



Obrázek A.3: Neopravitelný systém po kvantitativní analýze stromu



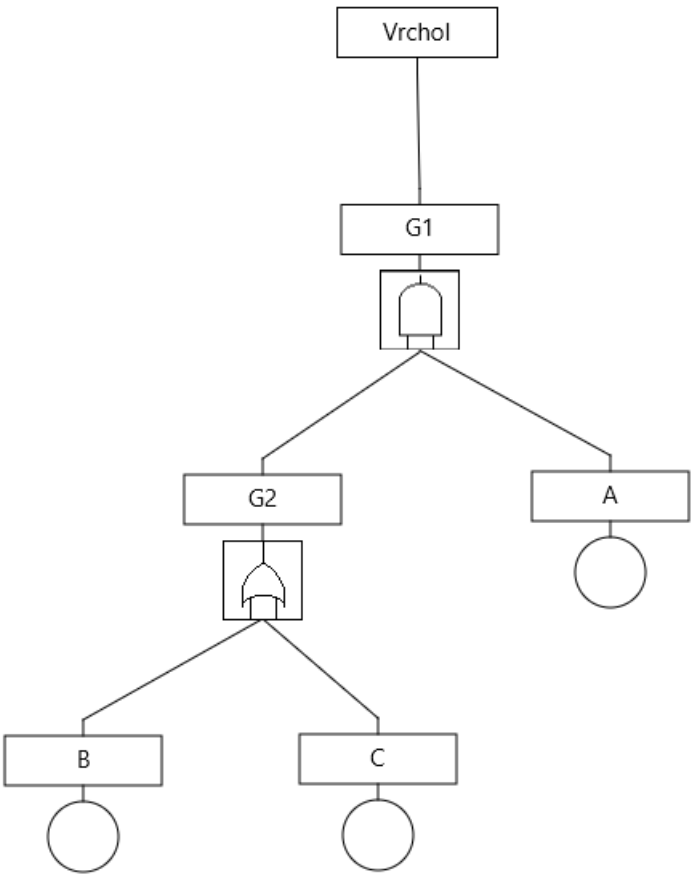
Obrázek A.4: Opravitelný systém po kvantitativní analýze stromu



Obrázek A.5: Další opravitelný systém po kvantitativní analýze stromu

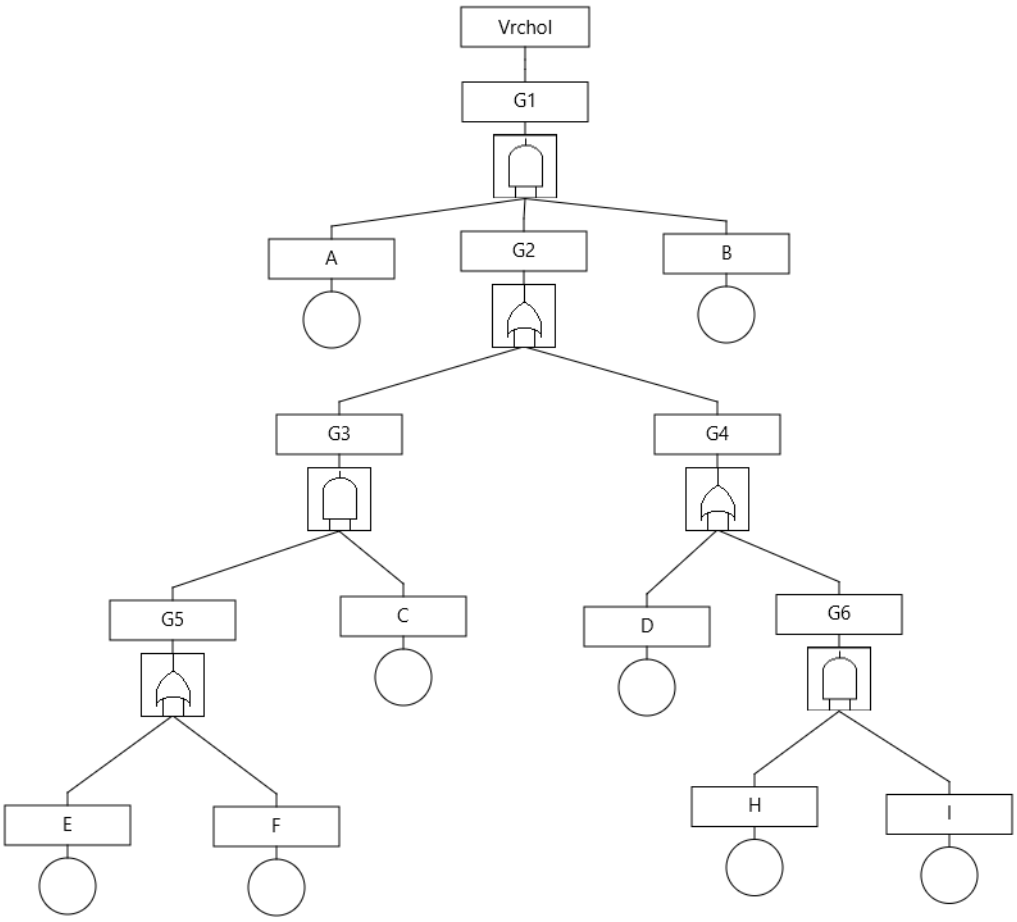
A.3 Testovací stromy poruch

A.3.1 První testovací strom poruch



Obrázek A.6: První testovací strom poruch

A.3.2 Druhý testovací strom poruch



Obrázek A.7: Druhý testovací strom poruch

B Ukázky kódu

B.1 Vyhledání základních událostí pomocí hledání do šířky

```
private ArrayList<IFTANode> getBasicEvents(IFTANode root)
   {
   Queue<IFTANode> nodes = new LinkedList<>();
   nodes.add(root);
   ArrayList<IFTANode> result = new ArrayList<>();

   IFTANode tmp = null;
   while(!nodes.isEmpty()){
      tmp = nodes.poll();
      if(!(tmp instanceof TopEvent || tmp instanceof
         TransferIn) && tmp instanceof AEvent){
         addWithDuplicityCheck(result, tmp);
      }
      if(tmp instanceof AGate){
         for(IFTANode n : ((AGate) tmp).getChildrens()){
            nodes.add(n);
         }
      } else {
         if(((AEvent) tmp).getChild() != null){
            nodes.add(((AEvent) tmp).getChild());
         }
      }
   }

   return result;
}
```

B.2 Ukázka metody textModify(String text) ve třídě PrimEventPane

```
@Override
public void textModify(String text) {
   Rectangle r = null;
   Text t = null;
   Line l = null;
   Circle c = null;

   //Cyklus pro nalezeni vsech dilcich casti
   //ze kterych je slozena znacka primarni udalosti
   for(Node n: this.getChildren()){
      if(n instanceof Rectangle){
         r = (Rectangle)n;
      }else if(n instanceof Text){
         t = (Text) n;
      }else if(n instanceof Line){
         if(((Line)n).getEndY() > 10) {
            l = (Line) n;
         }
      }else if(n instanceof Circle){
         c = (Circle)n;
      }
   }

   //Nastaveni pozadovaneho textu
   t.setText(text);
   //Nastaveni rozdeleni textu na vice radku pokud je
      sirsi nez ohraniceni
   if(t.getLayoutBounds().getWidth() > 90){
      t.setWrappingWidth(90);
   }

   //Nastaveni pozice textu v obdelniku
   t.setX((100-t.getLayoutBounds().getWidth())/2);
   //Nastaveni vysky obdelniku, ve kterem je text
      zobrazen
   r.setHeight(t.getLayoutBounds().getHeight()+10);
   //Uprava pozice znacky a car
   l.setStartY(r.getHeight()+10);
   l.setEndY(r.getHeight()+20);
   c.setCenterY(l.getEndY()+c.getRadius());
}
```

B.3 Metoda pro ukládání strom poruch do souboru ve třídě ObjectFileOperator

```
@Override
public void save(LinkedList<SerializablePane> panes,
   String filepath) throws IOException {

   //Vytvoreni souboru pro ulozeni FTA
   File f = new File(filepath);

   //Otevreni proudu pro ukladani dat do souboru
   FileOutputStream fos = new FileOutputStream(f);
   ObjectOutputStream oos = new ObjectOutputStream(fos);

   //zapis stromu do souboru
   oos.writeObject(panes);

   //Ukonceni zapisu do souboru
   oos.close();
   fos.close();
}
```

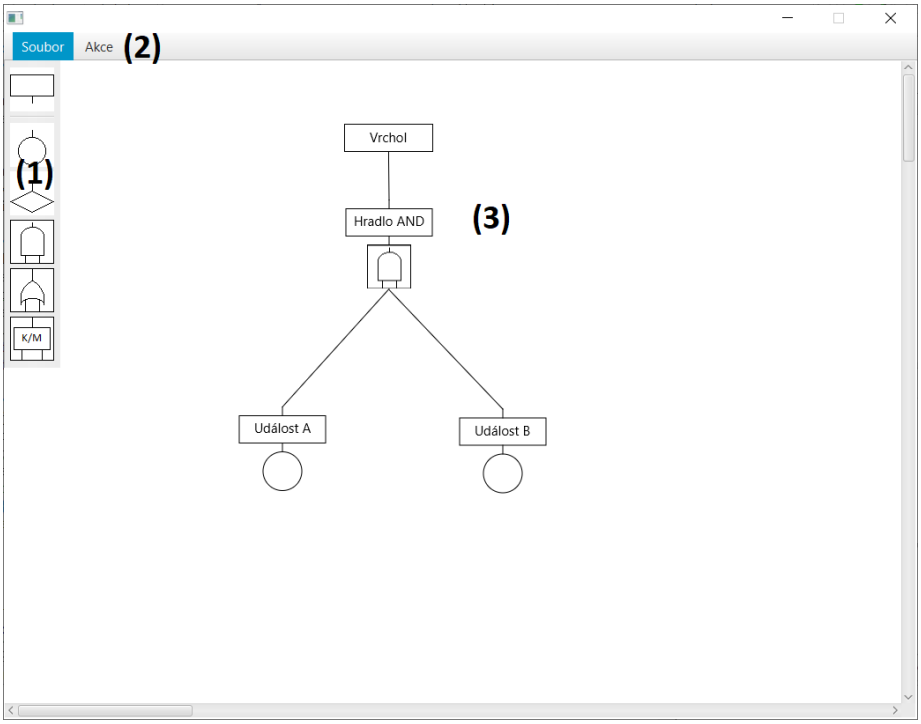
C Návod k aplikaci

C.1 Instalace

Pro spuštění aplikace je nutné mít nainstalovanou Javu verze 1.8 a vyšší. Nejnovější verzi Javy lze stáhnout z oficiálních stránek `https://java.com/en/`. Po instalaci Javy můžeme spustit program za pomoci spouštěcího souboru ***FTA.bat***.

C.2 Popis uživatelského rozhraní

C.2.1 Hlavní okno



(1) Panel nástrojů – obsahuje komponenty, ze kterých se sestavuje strom poruch.

(2) Menu – menu aplikace, umožňuje uložit nebo načíst projekt, vytvořit nový projekt, vyhodnotit sestavený strom poruch.

(3) Pracovní plocha – plocha, na které se sestavuje strom poruch pro analýzu.

C.2.2 Panel nástrojů

Panel nástrojů obsahuje komponenty pro sestavení stromu poruchových stavů. Pro přidání komponenty na pracovní plochu stačí na danou komponentu kliknout a držet levé tlačítko myši a přetáhnout ji na pracovní plochu.

C.2.3 Kontextové menu komponenty

Pro vyvolání kontextového menu stačí kliknout na vybranou komponentu pravým tlačítkem myši. Komponenty typu hradlo nelze duplikovat a zkopírovat.

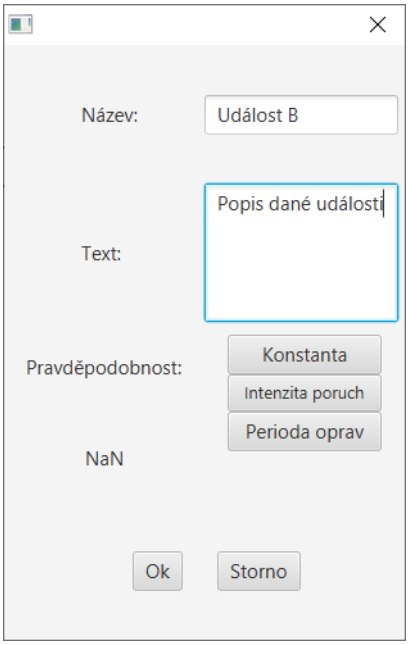
Obrázek C.1: Kontextové menu komponenty

- **Připojit** – po vybrání této položky a kliknutí na jinou komponentu v pracovní ploše se vytvoření propojení mezi nimi.
- **Upravit** – vyvolá nové okno pro úpravu komponenty.
- **Duplikovat** – vytvoří kopii komponenty tak, že změna v jedné z nich se projeví i u druhé.
- **Zkopírovat** – vytvoří novou komponentu, která má nastavené stejné hodnoty a parametry jako její předloha.
- **Smazat** – smaže komponentu z pracovní plochy i se všemi propojeními.

C.2.4 Úprava komponenty

Okno k úpravě komponenty vyvoláme tak, že z kontextového menu vybereme položku Upravit.

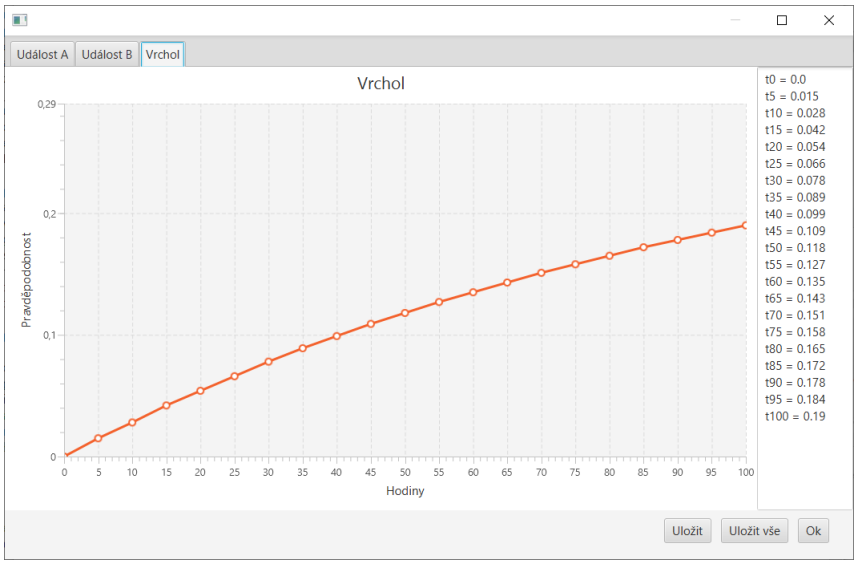
- **Název** – název dané komponenty.
- **Text** – detailní popis dané komponenty.



- **Pravděpodobnost** – zobrazuje s jakou pravděpodobností daná událost nastane. Číselná hodnota se zobrazí jen pokud byla události nastavena konstantní pravděpodobnost výskytu nebo po vykonání kvantitativního výpočtu.
- **Konstanta** – po stisknutí tohoto tlačítka vyskočí okno pro zadání konstantní pravděpodobnosti poruchy ve formě desetinného čísla v rozmezí od 0 do 1.
- **Intenzita poruch** – po stisknutí tohoto tlačítka vyskočí okno podobné oknu u konstanty pro zadání intenzity poruchy ve formě vědeckého zápisu čísla (např. 1.25E-2 = 0.0125).
- **Perioda oprav** – nastavuje se hodnota po jaké časové periodě se komponenta opraví.
- **M/N** – pouze u hradla M/N, nastavuje se kolik minimálně je potřeba porouchaných komponent u hradla, aby vznikla poruchová událost na hradle M/N. Lze nastavit jen v případě že hradlo má již k sobě připojené nějaké komponenty.

C.2.5 Okno analýzy stromu

Podle typu analýzy, kvalitativní nebo kvantitativní, vyskočí dialogové okno. Pro kvantitativní analýzu vyskočí okno vyobrazené níže. V tomto okně jsou zobrazeny



výsledky analýzy. Tlačítko **Uložit** slouží k uložení aktuálně zobrazeného grafu jako obrázek + zobrazené hodnoty u grafu se uloží do textového souboru. Tlačítko **Uložit vše** uloží všechny grafy, které během kvantitativní analýzy vznikly.

Podobné okno vyskočí i pro kvalitativní analýze, ale místo grafů se zobrazují vypsané minimální kritické řezy stromu poruch. Tlačítka **Uložit** a **Uložit vše** se v tomto okně nevyskytují.

D Obsah CD

- text bakalářské práce

  `BP_Vlastimil_Fengl.pdf`

- Latex projekt bakalářské práce

  `Latex_BP`

- Aplikace FTA editor

  `FTAEditorApp`

- Aplikace FTA Generator

  `FTAGeneratorApp`

- Zdrojový kód FTA editoru

  `FTA`

- Zdrojový kód FTA Generatoru

  `FTAGenerator`