TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Aplikace řešící spolehlivost systémů metodou stromu poruchových stavů (FTA)

Bakalářská práce

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

Autor práce: **Karel Drnec**
Vedoucí práce: Ing. Josef Chudoba, Ph.D.

Liberec 2020

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

Application for System Reliability Evaluation using Fault Tree Analysis (FTA)

Bachelor thesis

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

Author: **Karel Drnec**
Supervisor: Ing. Josef Chudoba, Ph.D.

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Aplikace řešící spolehlivost systémů metodou stromu poruchových stavů (FTA)

Jméno a příjmení: **Karel Drnec**
Osobní číslo: M16000020
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2019/2020**

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 práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] FUCHS, Pavel, Vališ DAVID, Chudoba JOSEF, Kamenický JAN a Zajíček JAROSLAV. *Řízení spolehlivosti* [online]. Liberec: Technická univerzita v Liberci, 2006 [cit. 2018-10-04].
[2] *ČSN EN 61025 (010676) Analýza stromu poruchových stavů (FTA)*. Praha: Česká technická norma, 2007.
[3] VESELY, W. E., F.F. GOLDBERG, N.H. ROBERTS a D.F. HAASL. *Fault Tree Handbook* [online]. Washington, D.C.: U.S. Nuclear Regulatory Commission, 1981 [cit. 2018-10-04]. Dostupné z: https://www.nrc.gov/docs/ML1007/ML100780465.pdf.
[4] www.weibull.com
[5] RAUSCHMAYER, Axel. *Speaking JavaScript: [an in-depth guide for programmers]*. Sebastopol, CA: O'Reilly Media, 2014. ISBN 9781449365035.
[6] BROWN, Ethan. *Web development with Node and Express*. Sebastopol, CA: O'Reilly, [2014]. ISBN 978-1491949306.

Vedoucí práce:	Ing. Josef Chudoba, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 17. října 2019

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

29. května 2020 Karel Drnec

Poděkování

Chtěl bych poděkovat panu Ing. Josefu Chudobovi, Ph.D. za odborné vedení, cenné rady, trpělivost a ochotu, kterou mi v průběhu této bakalářské práce věnoval. V neposlední řadě bych chtěl poděkovat svým blízkým za podporu.

Abstrakt

Tato bakalářská práce řeší spolehlivostní metodou FTA (analýza stromu poruchových stavů). Cílem práce v teoretické části je zpracování informací o metodě pro praktickou část práce. V praktické části je hlavním cílem udělat rešerši existujících softwarů a vytvořit uživatelsky přívětivý software pro metodu FTA, který umožní výpočet analýzy systému ze vstupních dat. Pro tvorbu aplikace je zvolen programovací jazyk JavaScript s technologií Node.js a webovým frameworkem Express.js. Dalším bodem v praktické části je ověření výpočtů vytvořené aplikace.

Klíčová slova: FTA, pravděpodobnost, porucha, JavaScript, Node.js, Express.js, bezpečnost

Abstract

This bachelor thesis is about reliability method FTA (Fault Tree Analysis). The goal of the thesis in a theoretical part is to process information about the method for practical part of the thesis. The main goal in the practical part is to research existing softwares and create an user-friendly software for FTA method that can calculates a system analysis from data inputs. Programming language JavaScript is chosen for app creation with Node.js technology and web framework Express.js. The next goal in the practical part is verification of calculations by created application.

Keywords: FTA, probability, failure, JavaScript, Node.js, Express.js, security

Obsah

Seznam obrázků

Seznam zkratek a použitého značení

FTA	analýza stromu poruchových stavů	
ETA	analýza stromu událostí	
BDD	Binary Diagram Decision	
FMEA	analýza druhů poruchových stavů a jejich důsledků	
t	čas	[*s*]
MTBF	střední doba mezi poruchami	[*hod*]
λ(t)	intenzita poruch	[hod^{-1}]
MTTR	střední doba do obnovy	[*hod*]
μ(t)	intenzita oprav	[hod^{-1}]
R(t)	pravděpodobnost bezporuchového provozu	[1]
F(t)	pravděpodobnost poruchy	[1]
A(t)	funkce okamžité pohotovosti	[1]
U(t)	funkce okamžité nepohotovosti	[1]
HTML	Hypertext Markup Language	
MVC	Model-View-Controller	
URL	Uniform Resource Locator	
NoSQL	nerelační databáze	
JSON	JavaScript Object Notation	
BSON	Binary JSON	
Blob	nestrukturovaný soubor binárních dat	

Úvod

Pojem spolehlivost je nesdílnou součástí každodenního života. Každá firma si na ní zakládá a před uvedením produktu či služby na trh předchází fáze testování spolehlivosti a odlaďování, které je často časově náročnější než samotné vytvoření prvotní verze. Nikdo nechce nabízet nekvalitní produkty či služby, protože by to neudělalo dobrou reklamu společnosti. Není to ale jen o reklamě. Spolehlivost je nejdůležitějším pojmem hlavně v takových odvětvích, jako jsou kosmonautika, letadlový, automobilový průmysl nebo energetika. Zde se spolehlivost často váže s bezpečností, která je v těchto odvětvích nejvíce brána v potaz a jakákoliv sebemenší chyba nějaké součástky by mohla znamenat případnou katastrofu.

Tato bakalářská práce se zabývá jednou z metod pro analýzu spolehlivosti a to metodou FTA (analýza stromu poruchových stavů), která se využívá pro vyhodnocování spolehlivosti složitých systémů. Nejdříve je v první části práce teoreticky popsána samotná metoda. Je zde zmíněna práce s analýzou, grafické značky využité v diagramu, vstupní či výstupní hodnoty nebo možnosti sestavení v systému. V rámci této práce jsou ve druhé části rozebrány existující komerční softwary, které se zabývají spolehlivostními metodami v reálném nasazení ve firmách po celém světě. Je zde popsáno rozhraní jednotlivých softwarů a jejich úskalí, práce při vytváření stromu poruchových stavů nebo možnosti výběru vstupních dat či zobrazení výsledků.

Ve třetí části práce je detailně popsána vytvořená aplikace. Je zde popsána struktura aplikace, algoritmus při tvorbě stromu poruchových stavů, vstupní a výstupní hodnoty, jednotlivé výpočty, generování náhodných systémů, možnosti načítání dat ze souboru a exportu dat. Součástí této části je celkové srovnání vytvořené aplikace s komerčními softwary zabývajícími se stejnou problematikou.

V závěrečné části této práce se nachází ověření správnosti výpočtu výsledků ve vytvořené aplikaci na základě testovaného příkladu složitějšího systému. V rámci této části je popsán princip ověření výpočtu a výsledné hodnoty jsou porovnány s hodnotami z vytvořené aplikace.

1 Analýza stromu poruchových stavů (Fault tree analysis)

1.1 Úvod

Analýza stromu poruchových stavů FTA je jedna z analytických metod pro analýzu spolehlivosti z kategorie deduktivních (řeší se shora dolů). Model je tvořen vrcholovou událostí, která se postupně dělí na jednotlivé primitivnější události způsobující danou vrcholovou událost. Vrcholovou událost obecně tvoří negativní jev (např. havárie nebo porucha). Tato metoda se zaměřuje na přesné určení příčin nebo jejich kombinací, které mohou vést k dané vrcholové události (jevu) a dále umožňuje snížit pravděpodobnost jeho výskytu.

Používá se pro vyhodnocování spolehlivosti složitých systémů a své využití nachází zejména v odvětvích jako je energetika, vesmírný výzkum či letectví. Většinou je využita jako preventivní metoda před poruchou, avšak často je používaná i po poruše. Analýzu lze provádět nezávisle na jiných analýzách spolehlivosti nebo společně s nimi. [1] [2]

FTA lze kombinovat s následujícími technikami analýzy spolehlivosti:

- FMEA (Failure Mode and Effect Analysis) – ČSN EN 60812,
- ETA (Event Tree Analysis) – ČSN EN 62502,
- Markovova analýza – ČSN EN 61165,
- BDD (Binary Diagram Decision),
- Blokový diagram bezporuchovosti – ČSN EN 61078.

První koncept byl vyvíjen firmou Bell Telephone Laboratories v souvislosti s vývojem startovacího systému rakety Minuteman pro Letectvo Spojených států amerických. Postupem času analýza nacházela čím dál více uplatnění v jaderné energetice a kosmonautice. V roce 1990 byla vydána mezinárodní norma, která popisovala principy a postupy pro tvorbu a vyhodnocování stromu poruchových stavů. V roce 1993 vyšla česká norma (ČSN IEC 1025 – Analýza stromu poruchových stavů), která byla později v roce 2007 rozšířena a poupravena. [3] [4]

1.2 Cíle metody FTA

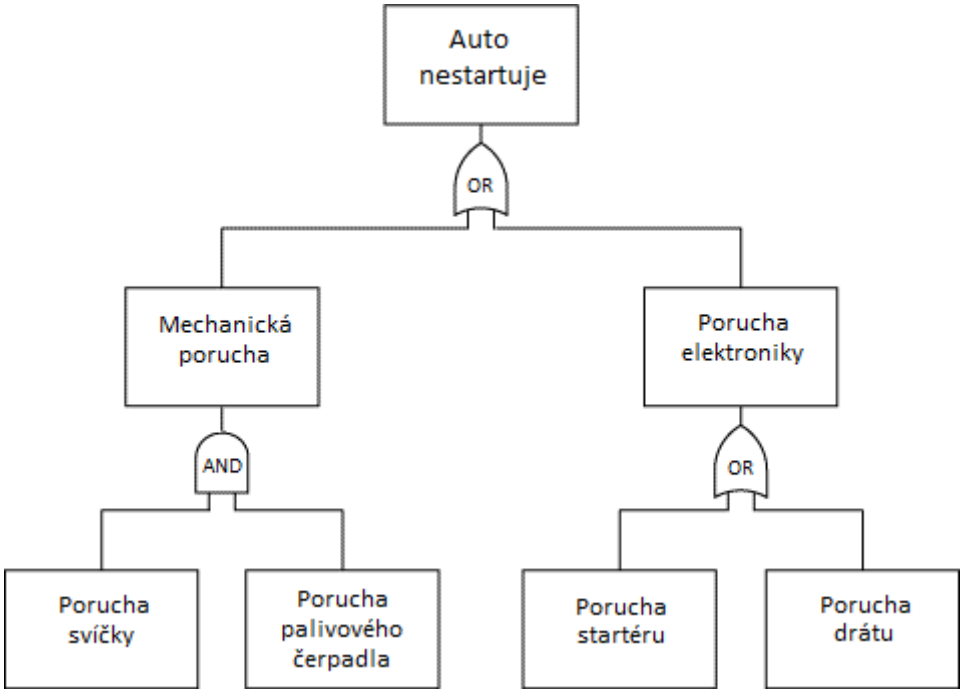
Hlavním cílem analýzy poruchových stavů je identifikace příčin nebo jejich kombinace vedoucí k vrcholové události. Mezi další významné cíle této metody patří:

- identifikování příčin nebo jejich kombinací, které způsobují vrcholovou – nežádoucí událost,
- zanalyzování návrhu a porovnání různých alternativ za účelem zlepšení spolehlivosti (bezporuchovosti, pohotovosti a udržovatelnosti) systému,

- identifikování příčin vzniku potenciálních poruch, které by nejvíce ovlivňovaly pravděpodobnost poruchy celého systému,
- hledání způsobů nápravných opatření a zjištění možných vylepšení bezporuchovosti,
- vypočítání pravděpodobnosti výskytu nežádoucích událostí. [1]

1.3 Fault tree diagram

V analýze stromu poruchových stavů Fault tree diagram (strom poruchových stavů) je grafická reprezentace podmínek nebo jiných faktorů způsobující vrcholovou událost. Patří mezi speciální orientované grafy. Strom se sestavuje od vrcholové události shora dolů pomocí grafických symbolů událostí, hradel a jednotlivých vazeb mezi nimi ve snadno pochopitelném tvaru, aby bylo možné strom co nejjednodušeji zanalyzovat a identifikovat jednotlivé příčiny problému nebo jejich kombinace a v případě potřeby mohlo dojít k přeskupení stromu pro lepší identifikaci. Ve stromu se stanovuje také pravděpodobnost výskytu ostatních událostí pro lepší určení poruchovosti komponent, které ovlivňují vrcholovou událost. [1] [3] [4]



Obrázek 1: Příklad stromu poruchových stavů

Hradla

Hradla jsou grafické značky sloužící k symbolickému značení logické vazby mezi výstupní událostí a odpovídajícími vstupy. Pro tuto logickou vazbu je použita Booleova algebra. Hradla se dělí na statická a dynamická, přičemž u statických hradel výstup nezávisí na pořadí výskytu vstupů. Mezi základní typy hradel patří hradla typu OR a AND. Ostatní typy hradel jsou pouze speciálními případy těchto dvou základních typů. Nejčastějším speciálním případem hradla je typ hradla K/N.

Naopak u dynamických hradel závisí na pořadí výskytu vstupů a jsou to například hradla typu prioritní AND nebo sekvenční. [1] [5]

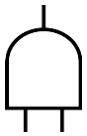
Hradlo OR

Hradlo OR značí logický součet. Výstupová událost nastane pouze tehdy, pokud nastane alespoň jedna ze vstupních událostí. [5]

Obrázek 2: Značení hradla OR

Hradlo AND

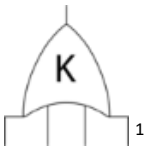
Hradlo AND značí logický součin. Výstupová událost nastane pouze tehdy, nastanou-li všechny vstupní události. [5]



Obrázek 3: Značení hradla AND

Hradlo K/N

Hradlo K/N je speciálním případem hradla vzniklého kombinací dvou předchozích. Výstupová událost nastane právě tehdy, pokud nastane K z N vstupů. [5]



Obrázek 4: Značení hradla K/N

Další používaná hradla

Další hradla používaná v analýze poruchových stavů jsou modifikací základních hradel OR a AND a jsou používaná pouze pro specifické, ne tak časté účely.

Hradlo XOR (exklusivní OR) je speciální případ hradla OR, kdy výstupová událost nastane pouze tehdy, pokud nastane právě jedna vstupní událost.

Prioritní AND hradlo je speciální případ hradla AND, kdy výstupová událost nastane pouze tehdy, nastanou-li všechny vstupní události ve specifickém stanovém pořadí. Pořadí je obvykle značeno v elipse umístěné napravo od daného hradla.

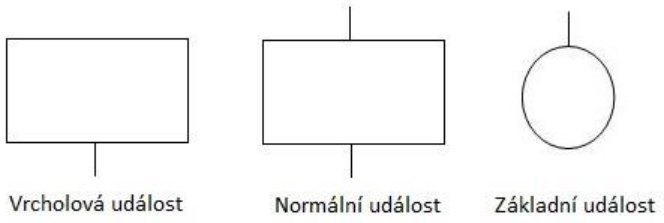
Sekvenční hradlo je rozšiřujícím případem hradla typu prioritní AND, kdy výstupní událost nastane pouze tehdy, nastanou-li všechny vstupní události v pořadí zleva doprava za podmínky, že existují více než dvě vstupní události.

[1] Někdy uváděno M/N místo K/N.

Hradlo INHIBIT je speciální typ hradla, kdy výstupová událost nastane pouze tehdy, nastanou-li všechny vstupní události a doplňková podmínková událost, která do hradla vstupuje externě. Jedná se vesměs o modifikaci hradla AND s podmínkovou událostí, která je zakreslená napravo od hradla v elipse. [1] [5]

Události

Události jsou grafické značky sloužící k značení výskytu určité podmínky či určitého děje. V následující části budou popsány základní druhy událostí. Mezi ně patří vrcholová událost (top event), normální událost (normal event) a základní událost (basic event). [1]



Obrázek 5: Značení událostí

Vrcholová událost

Vrcholová událost, někdy označována jako konečná, je událost, která je předmětem zájmu. Ve stromu poruchových stavů je vyobrazena jako tzv. kořen, pod kterým se strom dále rozvíjí pomocí logických hradel nebo normálních či základních událostí. Jedná se o výstup kombinací všech vstupních událostí. [1]

Normální událost

Normální událost, někdy označována jako mezilehlá, je událost, která není vrcholovou událostí, ani základní (primární) událostí. Od základní události se liší tím, že neobsahuje žádné vstupní hodnoty a je způsobena kombinací základních nebo dalších normálních událostí. [1]

Základní událost

Základní událost, někdy označována jako primární, je událost, která již není dál rozvíjena a nachází se na základní úrovni stromu. Obsahuje vstupní hodnoty, které jsou popsané dále. [1]

1.4 Kvalitativní analýza FTA

Kvalitativní analýzu FTA lze využít, pokud nejde odhadnout pravděpodobnost výskytu primárních událostí systému. Tyto události jsou označovány popisnou pravděpodobností výskytu jako „vysoce pravděpodobné“, „velmi pravděpodobné“, „středně pravděpodobné“, „velmi málo pravděpodobné“ atd.

Hlavním cílem kvalitativní analýzy je nalezení minimálního kritického řezu, podle kterého lze stanovit způsoby, jakými základní nebo primární (elementární) události ovlivňují vrcholovou událost. [1]

Kritický řez a minimální kritický řez

Podle ČSN EN 61025 (2007, s. 28) je kritický řez „skupina událostí, které při současném výskytu způsobí výskyt vrcholové události“ a minimální kritický řez „nejmenší množina událostí, které se nutně musí vyskytnout, aby způsobily vrcholovou událost.“

Vrcholová událost nenastane, pokud nenastane jakákoliv událost v minimálním kritickém řezu. [1]

1.5 Kvantitativní analýza FTA

Kvantitativní analýzu FTA lze použít v případě, kdy jsou známy jednotlivé pravděpodobnosti primárních událostí. Následně mohou být pravděpodobnosti výskytu všech základních událostí a vrcholové události vypočteny v souladu s modelem podle dané metody. Pro výpočet jsou používány tyto metody:

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

Cíl kvantitativní analýzy je zjistit ukazatele, které charakterizují a ovlivňují zkoumanou vrcholovou událost:

- pravděpodobnost výskytu vrcholové události v daném časovém intervalu,
- pravděpodobnost, že nenastane vrcholová událost v daném časovém intervalu,
- střední doba do prvního výskytu vrcholové události,
- střední počet výskytů vrcholové události v daném časovém intervalu. [1] [3]

1.6 Vstupní hodnoty

Do událostí nejnižší úrovně (základní události) vstupují různé typy parametrů. Mezi nejčastěji používané patří intenzita poruch (a oprav), pravděpodobnost bezporuchového provozu (pohotovost), či pravděpodobnost poruchy stanovena konstantní hodnotou.

Střední doba mezi poruchami

Střední doba mezi poruchami, označována zkratkou MTBF (Mean Time Between Failures), představuje očekávanou dobu provozu mezi poruchami. [6]

$$MTBF = \frac{\sum_{i=1}^{N} t_i}{N} \tag{1}$$

Intenzita poruch

Intenzita poruch, označována symbolem λ, je podle ČSN IEC 50 (1993, kapitola 191, s. 53) definována jako „limita poměru podmíněné pravděpodobnosti, existuje-li, že časový okamžik T vzniku poruchy objektu leží v daném časovém intervalu (t, $t + \Delta t$) k délce časového intervalu Δt, jestliže Δt se blíží nule, za podmínky, že na začátku časového intervalu je objekt v použitelném stavu.“ [6]

$$\lambda(t) = \frac{1}{MTBF}\ [hod^{-1}] \tag{2}$$

Střední doba do obnovy

Střední doba do obnovy, označována zkratkou MTTR (Mean Time To Repair), představuje očekávanou dobu do obnovy systému. [6]

$$MTTR = \frac{\sum_{i=1}^{N} t_i}{N} \tag{3}$$

Intenzita oprav

Intenzita oprav, označována symbolem μ, je podle ČSN IEC 50 (1993, kapitola 191, s. 57) definována jako „limita poměru podmíněné pravděpodobnosti, existuje-li, že zásah údržby po poruše skončí v časovém intervalu (t, $t + \Delta t$) k délce tohoto časového intervalu Δt, jestliže Δt se blíží nule, za podmínky, že tato operace neskončila do začátku časového intervalu.“ [6]

$$\mu(t) = \frac{1}{MTTR}\ [hod^{-1}] \tag{4}$$

1.7 Výstupní hodnoty

Pravděpodobnost bezporuchového provozu R(t)

Pravděpodobnost bezporuchového provozu, označována symbolem *R(t)*, je někdy nazývána jako funkce spolehlivosti a podle ČSN IEC 50 (1993, kapitola 191, s. 53) představuje „pravděpodobnost, že objekt může plnit požadovanou funkci v daných podmínkách v daném časovém intervalu (t_1, t_2).“ Pro nedegradující komponenty je nejčastěji popisována pomocí vzorce:

$$R(t) = e^{-\lambda t} \tag{5}$$

Toto tvrzení lze pouze uvážit za obecného předpokladu, že objekt se nachází ve stavu, kdy je schopen realizovat požadovanou funkci na začátku časového intervalu. [6]

Pravděpodobnost poruchy F(t)

V analýze stromu poruchových stavů se využívá opačné logiky, kde je třeba počítat s pravděpodobností poruchy. Ta je značená symbolem *F(t)* a jedná se o pravděpodobnostní doplněk k pravděpodobnosti bezporuchového provozu. [1]

$$F(t) = 1 - R(t) \tag{6}$$

Funkce okamžité pohotovosti A(t)

Funkce okamžité pohotovosti, označována symbolem *A(t)*, je podle ČSN IEC 50 (1993, kapitola 191, s. 48) definována jako „pravděpodobnost, že objekt je ve stavu schopném plnit v daných podmínkách a v daném časovém okamžiku požadovanou funkci, za předpokladu, že požadované vnější prostředky jsou zajištěny.“ [6]

$$A(t) = \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} \cdot e^{-(\lambda+\mu)t} \tag{7}$$

Funkce okamžité nepohotovosti U(t)

Funkce okamžité nepohotovosti, označována symbolem *U(t)*, je podle ČSN IEC 50 (1993, kapitola 191, s. 48) definována jako „pravděpodobnost, že objekt není ve stavu schopném plnit v daných podmínkách a v daném časovém okamžiku požadovanou funkci, za předpokladu, že požadované vnější prostředky jsou zajištěny.“ [6]

$$U(t) = 1 - A(t) \tag{8}$$

1.8 Vyhodnocení kvantitativní analýzy

Sériová konfigurace systému

Systém v sériové konfiguraci označuje takový systém, kde by porucha jakékoliv komponenty znamenala poruchu celého systému. V praxi to znamená, že komponenta 1 i komponenta 2 i ostatní, musí být provozuschopné, aby systém neselhal. V analýze stromu poruchových stavů je to takový systém, kde všechny tyto komponenty vstupují do hradla OR.

Pravděpodobnost bezporuchového provozu sériové konfigurace systému se spočítá pomocí vzorce:

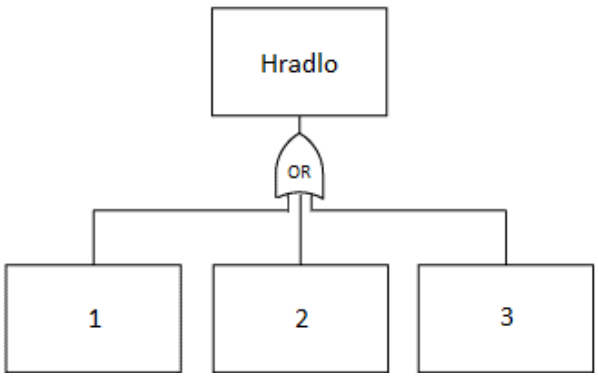
$$R_s(t) = R_1(t) \cdot R_2(t) \cdot \ldots \cdot R_N(t) = \prod_{i=1}^{N} R_i(t) \tag{9}$$

V analýze stromu poruchových stavů se využívá opačné logiky, kde je třeba počítat pravděpodobnost poruchy, která se spočte pomocí doplňku k pravděpodobnosti bezporuchového provozu. [1]

Pravděpodobnost poruchy sériové konfigurace systému se spočítá pomocí vzorce:

$$F_s(t) = 1 - \prod_{i=1}^{N} [1 - F_i(t)] \tag{10}$$

Obrázek 6: Sériová konfigurace o 3 komponentách



Obrázek 7: Strom v sériové konfiguraci o 3 komponentách

Paralelní konfigurace systému

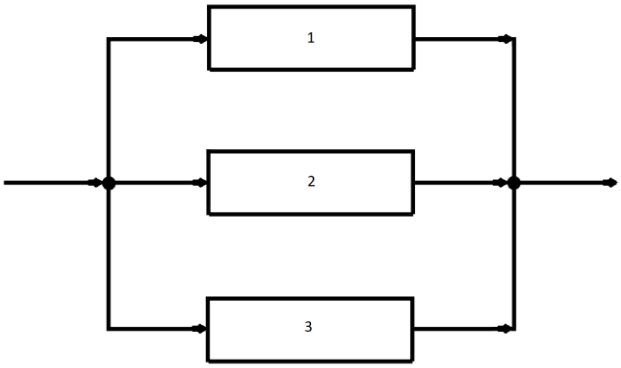
Systém v paralelní konfiguraci označuje takový systém, kde porucha celého systému nastane, pokud mají poruchu všechny jeho komponenty. Jedná se o zálohovaný systém, kde porucha jedné komponenty nezapříčiní selhání celého systému. V analýze stromu poruchových stavů je tento systém tvořen pomocí hradla AND. [1]

Pravděpodobnost bezporuchového provozu paralelní konfigurace systému se spočítá pomocí vzorce:

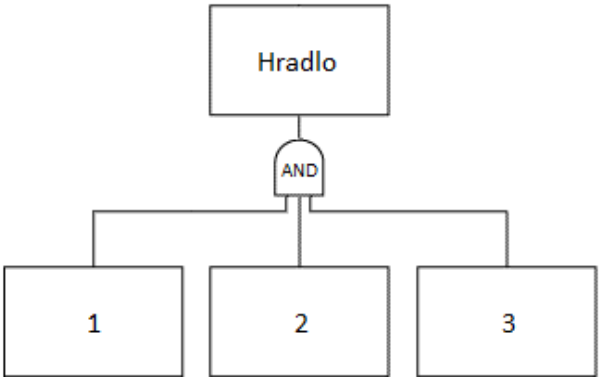
$$R_s(t) = 1 - \prod_{i=1}^{N}(1 - R_i(t)) \tag{11}$$

Pravděpodobnost poruchy paralelní konfigurace systému se spočítá pomocí vzorce:

$$F_s(t) = \prod_{i=1}^{N}[F_i(t)] \tag{12}$$



Obrázek 8: Paralelní konfigurace systému o 3 komponentách



Obrázek 9: Strom v paralelní konfiguraci o 3 komponentách

Konfigurace systému pomocí hradla K/N

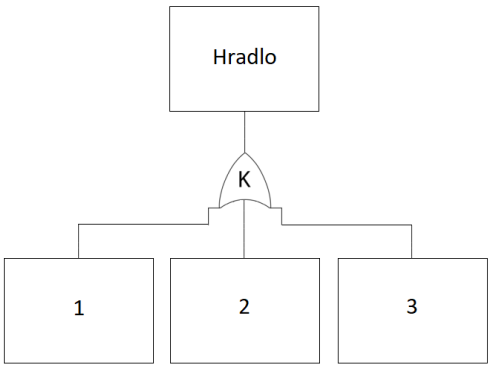
Systém v konfiguraci pomocí hradla K/N označuje takový systém, kde porucha celého systému nastane v případě, kdy selže více než K komponent z N možných. Jedná se o zálohovaný systém, který zůstane provozuschopný, pokud zůstane provozuschopných alespoň K komponent. [1]

Pravděpodobnost bezporuchového provozu konfigurace systému pomocí hradla K/N se spočítá pomocí vzorce:

$$R_s(t) = 1 - \sum_{i=0}^{k-1} \frac{n!}{i! \cdot (n-i)!} \cdot [R_0(t)]^i \cdot [1 - R_0(t)]^{n-i} \tag{13}$$

Pravděpodobnost poruchy konfigurace systému pomocí hradla K/N se spočítá pomocí vzorce:

$$F_s(t) = \sum_{i=0}^{k-1} \frac{n!}{i! \cdot (n-i)!} \cdot [1 - F_0(t)]^i \cdot [F_0(t)]^{n-i} \tag{14}$$



Obrázek 10: Strom v konfiguraci pomocí hradla K/N o 3 komponentách

2 Porovnání existujících softwarů

V následující části této práce jsou porovnány softwary zabývající se problematikou FTA. Jsou zde popsány výhody, nevýhody jednotlivých komerčních softwarů a práce s nimi při vytváření stromu poruchových stavů.

K porovnání byly vybrány následující komerční aplikace:

- Fault Tree Analyser (https://www.fault-tree-analysis-software.com/),
- ITEM ToolKit (https://www.itemsoft.com/item_toolkit.html),
- Isograph (https://www.isograph.com/software/reliability-workbench),
- ReliaSoft (https://www.reliasoft.com/products/reliability-analysis/blocksim),
- RAM Commander (https://aldservice.com/RAMS-Reliability-Availability-Maintainability-and-Safety-Software.html).

V závěrečné podkapitole se nachází krátká zmínka o vytvořené aplikaci v rámci této bakalářské práce. Podrobněji je aplikace popsána v následující části práce.

2.1 Fault Tree Analyser

Fault Tree Analyser je webová aplikace vytvořená společností ALD, která se zabývá oblastí spolehlivostního inženýrství a analýzy, bezpečnostní analýzy a řízení kvality. Mezi největší zákazníky této společnosti patří Národní úřad pro letectví a kosmonautiku (NASA) a výrobce civilních dopravních letadel Airbus. [7]

Rozhraní aplikace

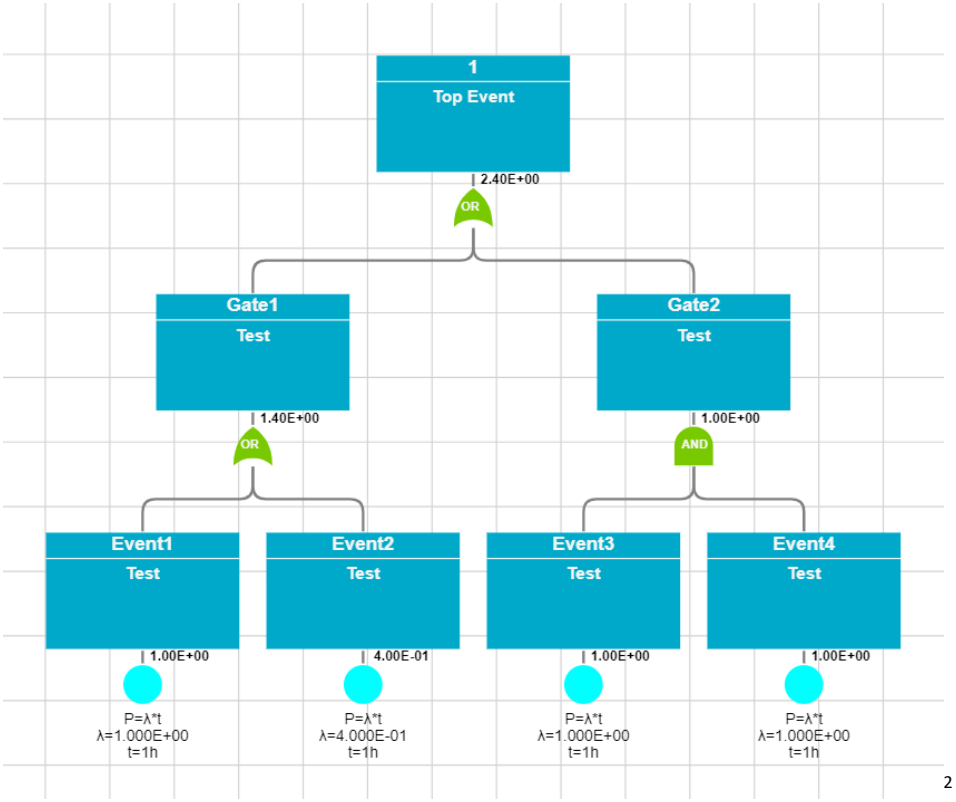
Při spuštění webové aplikace se zobrazí názorný příklad stromu poruchových stavů. Pro vytvoření vlastního stromu je třeba se registrovat. Po přihlášení do aplikace může uživatel pracovat na analýze vlastního problému.

Aplikace má hlavní nabídku ovládání umístěnou v horní části, která umožňuje uživateli pracovat na problému. Obsahuje záložky pro vytvoření nového projektu, načtení ze souboru, uložení do souboru, přidávání hradel a událostí a jejich případné modifikace nebo možnosti výpočtu poruchovosti systému.

Pro vytvoření nového projektu klikne uživatel na záložku *Fault Tree* a vybere položku *Create New*. Následně se zobrazí vyskakovací okno s nastavením vrcholové události, kde je potřeba nastavit unikátní název události. Uživatel může ještě poté vyplnit popis vrcholové události nebo změnit počet hodin trvání události, který je přednastavený na 0.

Pokud chce přidat nové logické hradlo nebo událost, stačí označit hradlo či událost, za kterými chce rozšířit větev stromu a dané hradlo nebo událost přidat. U logických hradel lze vybrat základní hradla typu OR, AND, K/N. U událostí lze přidat buď novou událost, nebo vybrat z existujících. Když uživatel nechce dále rozvíjet vybranou větev stromu, přidá novou událost, kde nastaví typ výpočtu a odpovídající vstupní hodnoty pro výpočet a tím se daná větev stromu uzavře. Lze vybrat výpočty typu konstantní pravděpodobnost, evidentní, latentní či MTTF. Událost lze také vybrat již existující z knihovny událostí.

Jestliže je strom zcela sestaven, lze vypočítat celkovou pravděpodobnost poruchového stavu pomocí *Probability Calculation* v záložce *Analysis* v hlavní nabídce aplikace. Po výpočtu se uživateli zobrazí detaily výpočtu pravděpodobnosti poruchového stavu, kde jednotlivé události se vstupními hodnotami jsou zaznamenány v tabulce pro lepší přehled.



Obrázek 11: Strom poruchových stavů vytvořený v Fault Tree Analyser

Hodnocení

Hlavním výhodou této aplikace je bezesporu to, že je bezplatná a online. Není potřeba instalace na zařízení. Nabízí interaktivní, jednoduché prostředí, ve kterém se uživatel snadno vyzná.

[2] Všechny události mají jako vstupní parametr počet poruch za hodinu.

Mezi nevýhody patří soukromí uživatele aplikace. Pracovat na svém vlastním problému lze pouze po registraci do systému. Společnost je vlastníkem dat a tak může někdy nastat problém, co se týče zakázek firem používající tento software. Aplikace taktéž někdy zahlásí chybu, kterou již blíže nespecifikuje a to může být pro uživatele dost matoucí.

2.2 ITEM ToolKit

ITEM ToolKit je software pro analýzu spolehlivosti od společnosti ITEM Software, která je dodavatelem softwarů analýz potřebných v průmyslových odvětvích po celém světě. Jedná se o placenou aplikaci, kde je možno si vyzkoušet demo verzi vázanou na autorizační klíč, který uživatel získá po vyplnění registračního formuláře. [8]

Rozhraní aplikace

Po instalaci softwaru lze v manuálu zjistit informace o demo verzi, která má následující omezení:

- nelze uložit vytvořený projekt,
- maximálně 5 projektů otevřených naráz,
- omezený počet bloků a komponent v systému (maximálně 25 bloků a komponent).

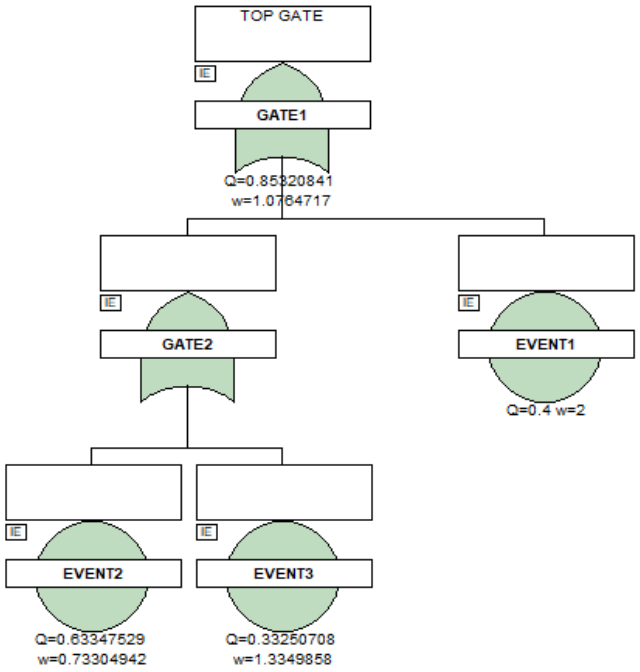
K vytvoření nového projektu v této aplikaci musí uživatel přidat z hlavní nabídky požadovaný modul. Na výběr má moduly FMECA, RBD, FTA, ETA a MKV. Po výběru modulu FTA se zobrazí pracovní plocha s vytvořenou vrcholovou událostí. Pokud uživatel chce přidat nové logické hradlo nebo událost, může buď kliknout na vybraný objekt a pomocí *Add* zvolit požadované nové hradlo či událost z nabídky, anebo může nový objekt vybrat z lišty objektů nad pracovní plochou aplikace a poté klikne na objekt ve stromu, za který se nové hradlo nebo událost zařadí. Uživatel má na výběr hradla typu AND, OR, K/N, XOR, prioritní AND, NOT nebo Inhibit.

Pro detailnější nastavení hradel a událostí, stačí označit daný objekt a zvolit z nabídky *Gate Parameters* nebo *Event Parameters*. Ve vyskakovacím okně lze pak parametry nastavit. Pro vložení vstupních hodnost musí uživatel v nastavení parametrů události do záložky *Failure* a zde zvolit typ vstupních hodnot a jednotlivé hodnoty vyplnit. Uživatel může vybrat parametry výpočtu události typu fixní, intenzita poruch, MTTF, nečinnost, aj.

Analýzu lze spustit označením vrcholové události a zvolením možnosti *Perform Gate Analysis* nebo z hlavní nabídky pomocí tlačítka *Start FT Analysis*. Po analýze se uživateli zobrazí vyskakovací okno s informací o celém systému, zda byl dobře sestaven a analýza proběhla v pořádku nebo obsahuje nějaké chyby způsobené například nenastavením vstupních hodnot událostí systému.

Nadále lze přidat obrázky do pracovního prostředí, kdy si může uživatel k událostem přidat i grafický popis a celé toto prostředí si pak vytisknout či exportovat do souboru. Také si může

zobrazit i manuál, kde je popsaná veškerá terminologie ohledně jednotlivých analýz zde zastoupených.



[3]

Obrázek 12: Strom poruchových stavů vytvořený v ITEM ToolKit

Hodnocení

Výhodou této aplikace je její propracovanost a všestrannost. Uživatel snadno a rychle přidá nové objekty do svého projektu a může vybranou analýzu zkombinovat s dalšími. Dalším pozitivem je možnost přidání obrázku do pracovní plochy.

Mezi nevýhody patří složitější nastavení událostí a jejich vstupních hodnot. Možnou nevýhodou je vypínání projektu a aplikace po delší nečinnosti ze strany uživatele. Toto ale může být z důvodu bezpečnosti, protože tahle aplikace je běžně nasazována v průmyslových odvětvích, kde na bezpečnosti analýz záleží.

2.3 Isograph

Isograph je software vytvořený stejnojmennou společností, která se zabývá spolehlivostními a bezpečnostními systémy. Společnost je certifikována mezinárodním standardem ISO 9001. Jedná se o placený software, kdy ho lze vyzkoušet po vyplnění formuláře na 7 dní. [9]

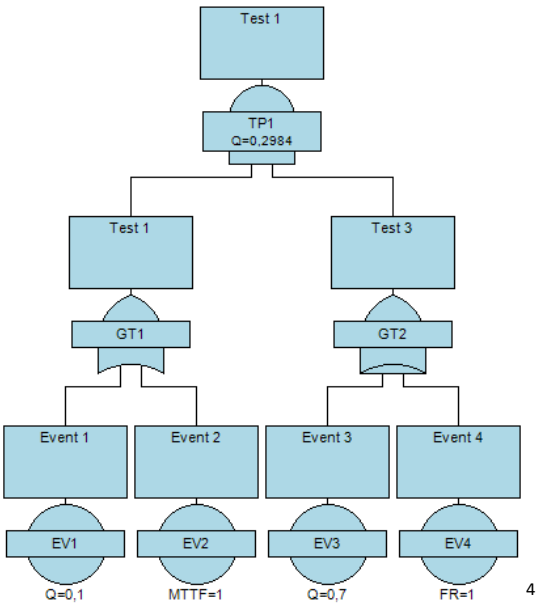
[3] U události 1 je nastaven vstupní parametr výpočtu na fixní. U událostí 2 a 3 je nastaven na Rate, kde byly doplněny hodnoty pro intenzitu poruch a intenzitu oprav.

Rozhraní aplikace

Po instalaci a spuštění aplikace se zobrazí vyskakovací okno s informacemi o bezplatné verzi a jejími omezeními, mezi která patří následující:

- omezený počet vytvořených objektů v systému (maximálně 60 hradel a 100 primárních událostí),
- nelze ukládat nebo exportovat projekt,
- aplikace může být zapnutá maximálně 4 hodiny v kuse.

Software není zaměřený pouze na analýzu FTA. Pro práci s FTA je zapotřebí vybrat daný modul z nabídky. Poté se zobrazí pracovní plocha, do které uživatel přidává objekty z hlavní nabídky, kde jsou přednastavené ikony pro vrcholovou událost, hradlo a událost. Dále se zde nachází ikony pro přidání poznámky nebo odkazu do pracovní plochy. Pokud uživatel klikne na ikonu vrcholové události, přidá se vrcholová událost s výchozími hodnotami. Pro přidání hradla nebo události se navolí objekt z nabídky a poté se klikne na objekt v ploše, za který se zařadí nové hradlo nebo událost.



Obrázek 13: Strom poruchových stavů vytvořený v aplikaci Isograph

Jestliže uživatel chce posunout objekt na jiné místo na ploše, označí ho a pomocí ikony v levém horním rohu objekt přesune na nové místo. Objekt nastaví označením a výběru možnosti *Edit Properties*. Pro vložení vstupních hodnot událostí zvolí v nastavení vlastností *Create a new local failure model*. Zobrazí se okno pro nastavení typu výpočtu a hodnot.

[4] U událostí 1 a 3 jsou nastaveny konstantní pravděpodobnosti. Událost 2 má parametr střední dobu do poruchy a událost 4 intenzitu poruch.

Po sestavení systému si lze ověřit správnost sestavení pomocí tlačítka *Verify Model Integrity* v hlavní nabídce a pak stačí zvolit *Perform Analysis* pro spuštění analýzy.

Hodnocení

Výhodou této aplikace je rychlost jak vytvoření stromu poruchových stavů, tak i analýzy systému. Další výhodou je jednoduché nastavení vstupních parametrů události.

Mezi nevýhody patří složitější uživatelské prostředí pro začínajícího uživatele a horší přemisťování objektů. Další možnou nevýhodou jsou omezení v bezplatné verzi, které ale v placené nejsou.

2.4 ReliaSoft

ReliaSoft je software vytvořený stejnojmennou společností zabývající se spolehlivostí, která od roku 2015 spadá pod společnost HBM Prenscia. Software lze vyzkoušet po vyplnění registračního formuláře, kdy je následně zaslán odkaz na stažení demo verze a autorizační klíč platný po dobu 30 dní. [10]

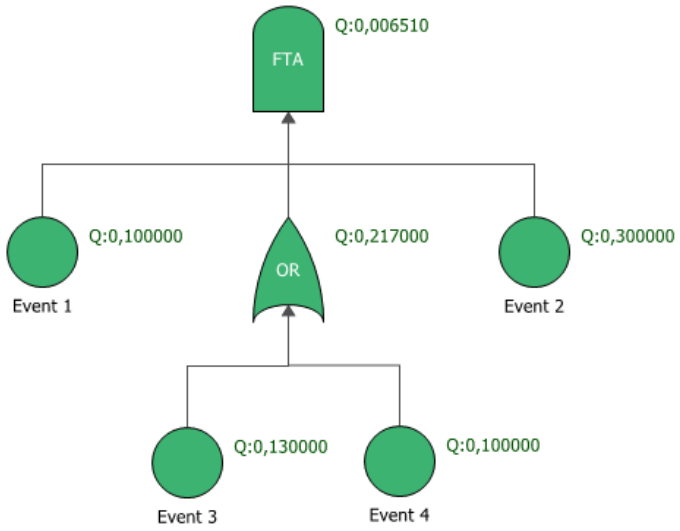
Rozhraní aplikace

Po zapnutí aplikace se spustí launcher s nabídkou modulů pro různé analýzy. Pro FTA se zvolí modul *BlockSim*. Zobrazí se nabídka pro nastavení pracovního repozitáře. Po nastavení se načte uživatelská plocha. K vytvoření projektu musí uživatel po levé straně zvolit z nabídky záložku *Fault Trees* a zvolit možnost *Add Analytical Fault Tree*. Prostředí se upraví a v hlavní nabídce se zobrazí možnosti pro přidávání hradel nebo událostí. Lze přidat hradla typu AND, OR, K/N, NAND, NOR, prioritní AND, aj. Událost lze přidat typu základní, nerozvíjená, spouštěcí nebo podmínková.

Při tvorbě stromu se musí vytvořit hradlo, které bude symbolizovat vrcholovou událost a které se bude rozvíjet. K rozvíjení objektu stačí objekt označit a zvolit možnost *Add Event* nebo *Add Gate*. Nově přidané objekty se umístí buď na sebe a poté se musí manuálně přemístit nebo lze zaškrtnout v hlavní nabídce možnost *Auto Arrange* a objekty se automaticky rozmístí do plochy po úpravě.

Pro nastavení objektu stačí objekt označit a vybrat možnost *Block Properties*. K přidání vstupních hodnot události se v nastavení zvolí možnost *Model – Failure*, kde vstupní hodnotu může tvořit již existující model, konstantní pravděpodobnost nebo hodnota vypočítaná z definovaných funkcí,

kam se pouze doplní odpovídající hodnoty. Analýzu uživatel spustí zvolením možnosti *Analyze* z hlavní nabídky pod záložkou *Analysis*.



[5]

Obrázek 14: Strom poruchových stavů vytvořený v aplikaci ReliaSoft

Hodnocení

Tato aplikace je nejvíce komplexní v porovnání s ostatními. Hlavní výhodou je uživatelské prostředí, které je intuitivní a propracované. Další výhodou je podpora ze strany firmy a komunikace s ní. Firma se také sama zajímala o pocity uživatele při práci se softwarem a byla ráda za recenzi. Autorizační klíč lze využít i k dalším modulům této aplikace.

Mezi možné nevýhody patří instalace aplikace. Během instalace znovu požaduje informace o uživateli, kdy po vyplnění na zadané telefonní číslo začne volat číslo ze státu mimo Evropskou unii.

2.5 RAM Commander

RAM Commander je software vytvořený společností ALD. Na rozdíl od webové aplikace Fault Tree Analyser, se zde jedná o plnohodnotný software, který se zabývá i dalšími analýzami. Pro získání softwaru stačí vyplnit registrační formulář. Vzápětí si lze demo ihned stáhnout a vyzkoušet. [11]

[5] Všechny vstupní události ve stromu mají nastavenou konstantní pravděpodobnost.

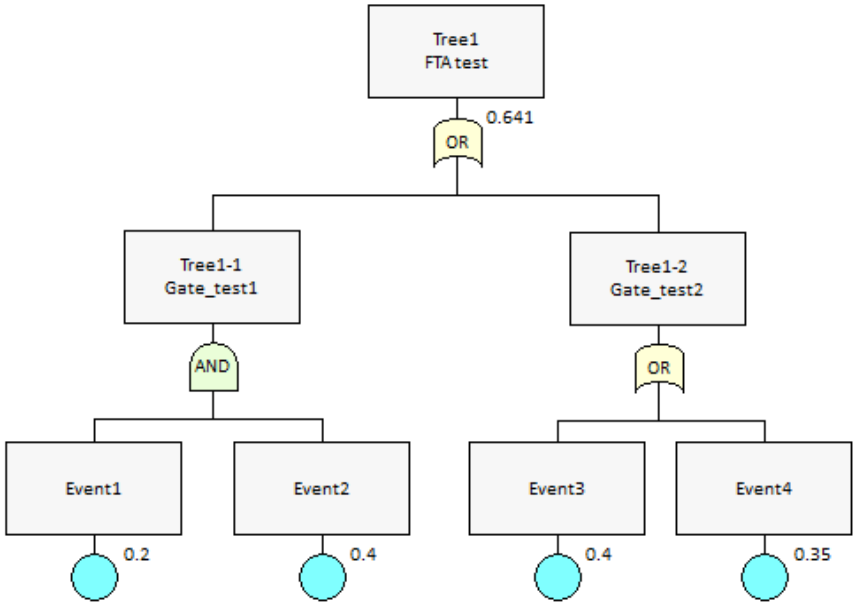
Rozhraní aplikace

Po instalaci softwaru lze v manuálu najít informace o edicích. Demo verze má následující omezení:

- lze mít uložené pouze 3 systémy v aplikaci,
- maximálně 15 hradel v systému,
- omezené vstupní parametry.

Po spuštění aplikace se zobrazí vyskakovací okno s dostupnými moduly. Následně po výběru modulu *Fault Tree Analysis* lze v novém okně vytvořit novou vrcholovou událost nebo zobrazit vytvořené systémy k ukázce pro začínající uživatele. K vytvoření nového projektu se zde vyplní jméno vrcholové události a následně potvrdí. Vrcholová událost se umístí do pracovní plochy.

Pro přidání hradla či události stačí daný objekt zvolit a vybrat možnost *Add Element*. Zobrazí se společné okno jak pro hradla, tak i pro události. Typ objektu lze nastavit v možnosti *Element type*. Po zvolení typu se okno aktualizuje a lze poté doplnit příslušné hodnoty pro daný objekt. Hradla jsou zde na výběr typu OR, AND, NOR, NAND, XOR, K/N, prioritní AND či sekvenční. Událost lze přidat i z knihovny, kde je seznam již nastavených událostí a to pomocí možnosti *Add basic event from library*. Objekt se po nastavení zařadí za svého předka.



[6]

Obrázek 15: Strom poruchových stavů vytvořený v aplikaci RAM Commander

[6] Události mají nastavenou konstantní pravděpodobnost.

K provedení analýzy stačí zvolit z hlavní nabídky možnost *Calculation* a zobrazí se hlášení s výsledky nebo možnými chybami v sestavení systému. Projekt lze uložit, avšak v této bezplatné verzi lze mít k dispozici pouze 3 uložené projekty.

Hodnocení

Hlavní výhodou této aplikace je rychlost provedení výpočtu a možnosti jeho nastavení. Další výhodou je rychlost dodání bezplatné verze, kdy stačí pouze vyplnit příslušný registrační formulář a ihned lze aplikaci stáhnout a vyzkoušet.

K nevýhodám lze zařadit špatné automatické vkládání nových objektů, kdy se při větším množství vkládají přes sebe. Uživatel poté musí objekty rozmístit manuálně. Možnou nevýhodou pro začínající uživatele může být společné nastavovací okno jak pro hradla, tak i pro události.

2.6 Vlastní aplikace

Vlastní aplikace vznikla v rámci této bakalářské práce na téma Analýza stromu poruchových stavů (FTA). Jedná se o webovou aplikaci (https://fta-app.herokuapp.com/), do které lze přistoupit pouze pod vytvořeným uživatelským účtem, který se vytvoří pomocí registračního formuláře. Rozhraní aplikace je podrobně popsáno v kapitole 3.5. Hlavní výhodou této aplikace je jednoduchá obsluha pro nenáročného uživatele, který potřebuje pro řešení problému pouze analýzu FTA. V kapitole 3.6 je aplikace srovnána s komerčními softwary, které byly probrány v této části práce.

3 Aplikace

V následující části práce je popsána vytvořená aplikace (https://fta-app.herokuapp.com/). Nejprve jsou zde vysvětleny informace o aplikaci a použité technologie. Následuje popis struktury, postup výpočtu analýzy, zabezpečení a samotné ovládání aplikace. Na závěr kapitoly je vytvořená aplikace porovnána s komerčními softwary, které byly podrobně rozebrány v předchozí části práce (viz Kapitola 2).

3.1 O aplikaci

Jedná se o webovou aplikaci vytvořenou ve značkovacím jazyce HTML a programovacím jazyce JavaScript za pomoci prostředí Node.js (https://nodejs.org/en/) a jeho webového aplikačního frameworku Express.js (https://expressjs.com/). Aplikace pracuje na základě architektonického návrhového vzoru Model View Controller (zkráceně MVC), který se skládá ze tří funkčních komponent:

- Model – datová struktura, se kterou aplikace pracuje,
- View – „pohled“, reprezentace dat do uživatelského prostředí,
- Controller – tzv. řadič, který zajišťuje komunikaci mezi modely a pohledy.

Komunikaci mezi webovou aplikací a controllerem zajišťuje tzv. router, který přijímá požadavky (zpracuje URL) a podle parametrů určí, který controller má být použit. Tento controller pak podle svých parametrů zavolá daný model kvůli datům, které pak vyšle do připravené šablony zpět na view (uživatelské prostředí aplikace).

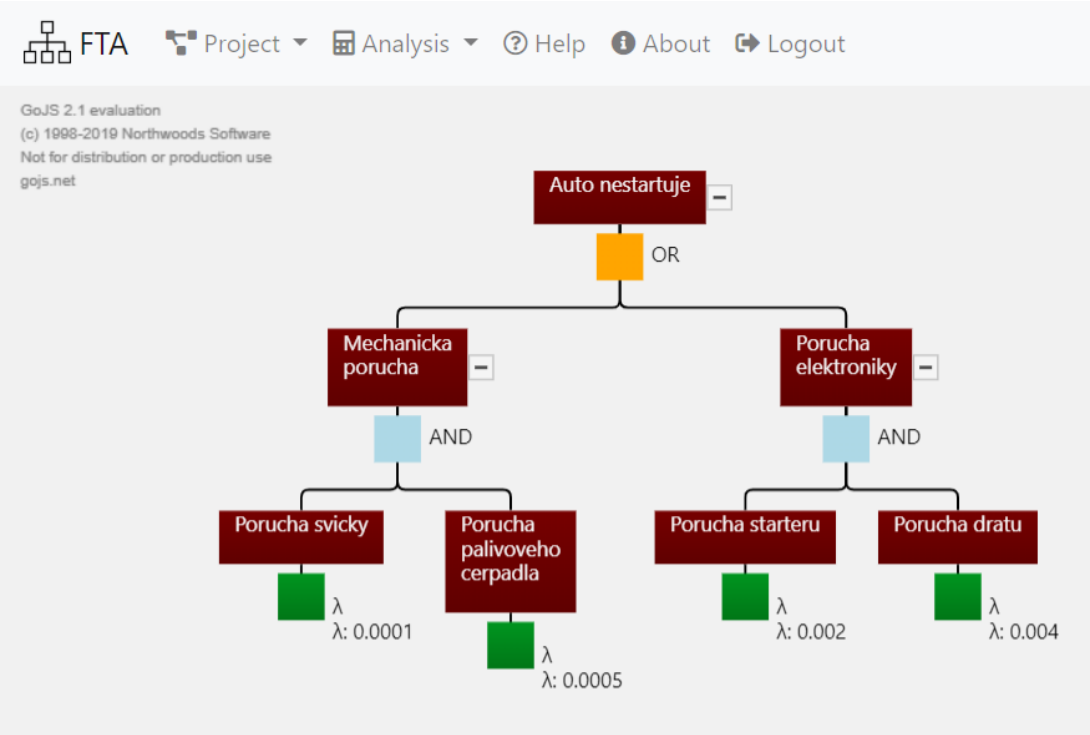
Vzhled webové aplikace zajišťuje open-source knihovna Bootstrap (https://getbootstrap.com/), která obsahuje nadefinovanou sadu nástrojů pro úpravu typografie a vzhledu webové stránky. Pro strukturovanost webové aplikace byl použit šablonový systém Handlebars (https://www.npmjs.com/package/handlebars), který umožňuje tvorbu šablon, ze kterých mohou webové stránky dědit společné údaje. Handlebars umožňuje tvorbu znovupoužitelných komponent (tzv. partials). V aplikaci pomocí partials byly vytvořeny 2 navigační lišty hlavní nabídky aplikace. První lišta obsahuje veškeré ovládání aplikace (nový projekt, export, analýza, aj.), druhá byla vytvořena za účelem zachování loga v ostatních stránkách aplikace, které slouží po kliknutí jako návrat do hlavního pracovního prostředí aplikace.

Pro uchování uživatelských dat byla použita multiplatformní dokumentová databáze MongoDB (https://www.mongodb.com/). MongoDB patří mezi NoSQL databáze, kde se nevyužívá klasických tabulkových schémat pro práci s daty. Data jsou tvořeny podle vytvořených schémat a poté jsou ukládány do dokumentů ve formátech JSON (popřípadě BSON). Databáze v aplikaci uchovává uživatele, jejich přihlašovací údaje a data. Webhosting vytvořené aplikace zajišťuje

cloudová platforma Heroku (https://devcenter.heroku.com/), která nabízí hosting aplikace zdarma. [12] [13]

Zobrazení diagramu

K zobrazení stromu poruchových stavů v grafické podobě byla zvolena JavaScriptová knihovna GoJS (https://gojs.net/latest/index.html), která nabízí stromové reprezentace grafů, jako jsou například UML diagramy, rodokmeny, turnajoví pavouci, stromy poruchových stavů, aj. Pro splnění účelu šlo alternativně zvolit i knihovnu Treant.js (https://fperucic.github.io/treant-js/).



Obrázek 16: Strom poruchových stavů v prostředí vytvořené aplikace

Knihovna GoJS byla zvolena, protože obsahuje nadefinovaný model stromu poruchových stavů, který lze upravit nebo rozšířit dle potřeb. Dalšími výhodami knihovny jsou:

- dobře popsaná dokumentace s názornými příklady,
- obsáhlejší (neobsahuje pouze čistě stromové struktury grafů),
- více objektová, vzhledově intuitivnější.

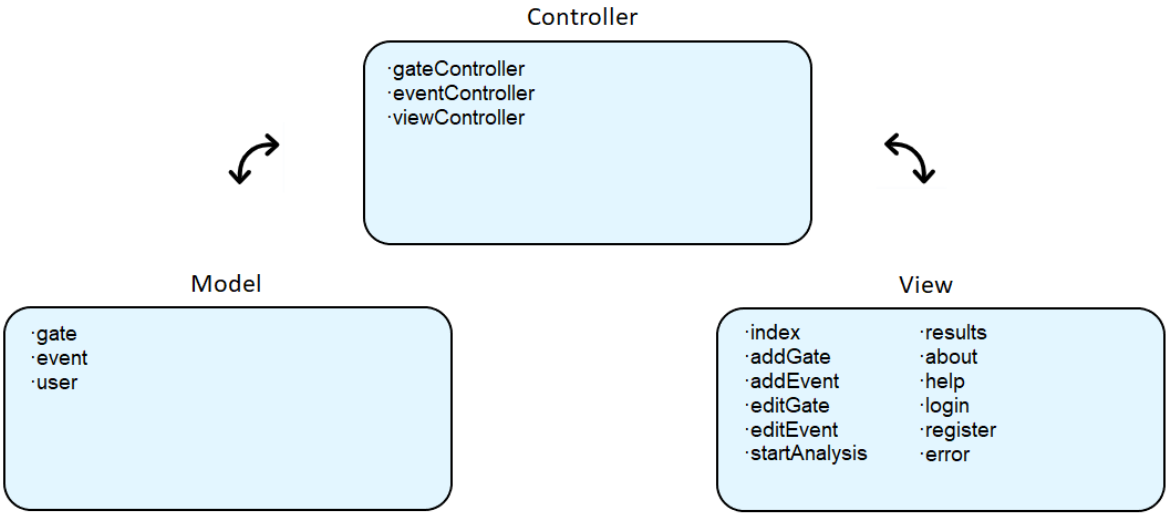
Knihovnu lze použít bezplatně pro soukromé účely, kdy se informace o knihovně zobrazí v levém horním rohu prostředí. Pro komerční účely lze vybrat z následujících licencí:

- individuální – cena 3 495 $,
- týmová – cena 6 990 $ pro maximálně 3 developery,
- skupinová – cena 9 950 $ pro neomezený počet developerů.

Všechny licence jsou platné po dobu tří let a obsahují podporu a aktualizace ze strany společnosti na 1 rok. [14] [15]

3.2 Struktura vytvořené aplikace

V této podkapitole je detailněji popsána struktura vytvořené aplikace. Na následujícím obrázku je zobrazen návrh aplikace podle návrhového vzoru MVC, který byl takto i implementován.



Obrázek 17: Návrh aplikace v MVC

Pro tvorbu objektových schémat pro databázi MongoDB byl použit nástroj Mongoose (https://mongoosejs.com/), který vytvořené schéma validuje a umožňuje jeho následné použití v Controlleru. V aplikaci jsou definovaná tato schémata:

- User – uživatelské jméno, email, heslo, ID uživatele,
- Gate – název, typ hradla, hodnota, popis, ID hradla, ID uživatele, ID předka,
- Event – název, typ události, hodnoty, popis, ID události, ID uživatele, ID předka.

V MongoDB se ID objektů generují automaticky a není je potřeba definovat do schémat jako další údaj objektu. Pokud ale je zapotřebí mít vlastně definované ID, musí se v schématu nastavit hodnota pro ID na *false* a specifikovat vlastní určení ID objektů. Pro komunikaci mezi daty a pohledy jsou definovány 3 Controllery:

- eventController - uložení, získání události podle ID, získání všech událostí, aktualizace, odstranění,
- gateController – uložení, získání hradla podle ID, získání všech hradel, aktualizace, odstranění,
- viewController – získání všech hradel a událostí (pro tvorbu diagramu), odstranění všech hradel a událostí.

View Controller byl vytvořen za účelem asynchronního získání a odstranění jak hradel, tak událostí z databáze. [16]

Pro komunikaci mezi webovou aplikací a controllery byly vytvořeny následující routery:

- Index – přesměrování pro zobrazení dat o aplikaci, manuálu, analýzy, výsledků a informací o aplikaci,
- Users – přihlášení, registrace a odhlášení uživatelů,
- Gates – směrování pro tvorbu, editaci a odstranění hradel,
- Events – směrování pro tvorbu, editaci a odstranění událostí.

Použité datové struktury pro sestavení systému

V aplikaci se údaje o vytvořeném systému uchovávají v objektech typu JSON. JSON (JavaScript Object Notation) je odlehčený datový formát nezávislý na počítačové platformě. Vstupem formátu typu JSON je libovolný datový typ nebo struktura (String, Integer, Boolean, aj.). V aplikaci jsou do objektů typu JSON zapsána data v podobě klíče a jeho hodnoty. [17]

Data o hradlech a událostech uložených v databázi, se získávají pomocí příslušných Controllerů do uživatelského prostředí aplikace. Získaná data jsou v datovém typu String a obsahují speciální entity znaků, které se zobrazí pouze v HTML (např. uvozovky, mezery, aj.). V JavaScriptu se místo těchto entit zobrazí jejich kód. Pro použití dat v JavaScriptu je zapotřebí získaná data parsovat[7]. Nejdříve se pomocí funkce *replace()* nahradí kód entity za textový řetězec, který označuje. Následně je použita funkce *JSON.parse()*, která textový řetězec převede do objektu typu JSON.

Upravená data jsou vložena do objektu typu JSON s názvem *objects_cal_array*, který obsahuje jak informace o hradlech, tak i o událostech. Objekt *objects_cal_array* obsahuje následující klíče:

- key – ID objektu (ID objektu z databáze),
- name – název objektu,
- type – typ objektu (gate nebo event),
- gateType – typ hradla (u události eventType – typ události),
- parent – ID objektu, jehož je potomkem.

U hradel se navíc uchovává údaj *number*, který označuje parametr K u hradla K/N. V případě hradla AND nebo OR je tento údaj nastaven na výchozí hodnotu 0. U událostí se uchovává navíc parametr *values*, který obsahuje nastavené vstupní hodnoty.

[7] Jedná se o převod z textové podoby na jiný datový typ (např. číslo).

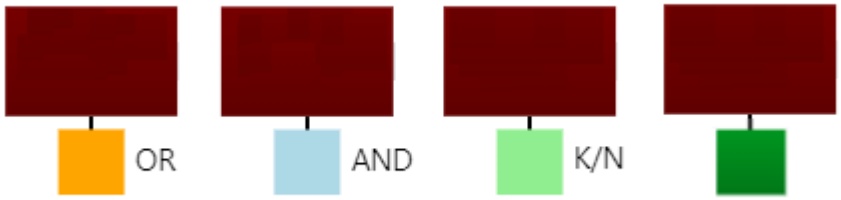
Pro přístup k objektu *objects_cal_array* na dalších stránkách (např. nastavení analýzy), je tento objekt po naplnění hodnotami uložen do tzv. *sessionStorage*. Uložiště *sessionStorage* uchovává data v rámci dané session, která v aplikaci je za účelem rozeznání dat různých uživatelů a vzniká přihlášením uživatele. Nový prvek se do *sessionStorage* přidá pomocí funkce *setItem()*, do které jako parametry vstupují název objektu a data v podobě textového řetězce (String). Pro uložení nového prvku se musí jiný datový formát převést na textový řetězec. Pro objekt *objects_cal_array*, který je typu JSON, se využívá funkce *JSON.stringify()*, která objekt typu JSON převede na textový řetězec. Data ze *sessionStorage* se získají pomocí funkce *getItem()*, do které vstupuje jako parametr název, pod kterým byla data do *sessionStorage* uložena. Po odhlášení, kdy se session z webového prohlížeče odstraní, se odstraní i veškerá data uložená v *sessionStorage*. [18]

Formát dat pro vytvoření diagramu

K zobrazení diagramu pomocí knihovny GoJS se data potřebná pro graf uchovávají v objektu typu JSON s názvem *jsonArrDiagram*. Objekt *jsonArrDiagram* obsahuje následující klíče:

- key – ID objektu (ID objektu z databáze),
- text – text objektu (složený z názvu objektu a popisku),
- figure – typ schématu, který se má vykreslit,
- choice – text, zobrazující se napravo od objektu (obsahuje informace o vstupních datech, typu dat a výsledky analýzy systému),
- parent – ID objektu, jehož je potomkem.

U parametru *figure* jsou na výběr základní hradla (OR, AND, K/N) a základní událost (Event). Na následujícím obrázku (Obrázek 18) je grafická intepretace dostupných objektů v diagramu.



Obrázek 18: Hradla OR, AND, K/N a základní událost

Přístup k datům v diagramu

K datům v aplikaci lze přistoupit pomocí grafických objektů v diagramu. Pomocí možnosti *part* se získá objekt v diagramu označený uživatelem. Z takto získaného objektu se získají data objektu zvolením možnosti *data*. Následně lze přistoupit již ke klíčům v objektu *jsonArrDiagram*

pomocí zvoleného klíče. Ve vytvořené aplikaci se pro práci s daty zejména využívají klíče *key* a *figure* z *jsonArrDiagram*. Objekty se v diagramu ovládají pomocí vytvořeného *ContextMenu*, které na základě klíče *figure* nabízí 2 různá zobrazení. Pokud se jedná o hradlo (*figure* má hodnotu OR, AND nebo K/N), zobrazí se nabídka na přidání nového hradla (*Add Gate*), přidání nové události (*Add Event*), editace označeného hradla (*Edit*) a smazání vybraného hradla (*Delete*). Pokud zvolený objekt v diagramu je událost (*figure* má hodnotu Event), zobrazí se pouze nabídka na editaci (*Edit*) a smazání označené události (*Delete*). Klíč objektu *key* se využívá pro následné úpravy nebo přidávání objektů. Při zvolení možnosti pro přidání nebo editaci hradla (události), se přes klíč *key* zjistí ID zvoleného objektu, které je následně vloženo jako parametr ID do URL stránky pro přidání nového hradla (události) nebo jako parametr ID do URL stránky pro editaci zvoleného hradla (události).

Smazání objektu v diagramu

Označený objekt v diagramu (hradlo nebo událost) lze smazat pomocí možnosti *Delete* v nabídce *ContextMenu*. Ve vytvořené funkci *deleteNode()* je posuzován klíč objektu *figure*, zda se jedná o hradlo (*figure* má hodnotu OR, AND nebo K/N) nebo událost (*figure* má hodnotu Event). Pokud zvolený objekt je typu hradlo, je nadále posuzováno, zda na něj navazují další hradla nebo události ve funkci *makeArrayToDelete()*, která jako vstupní parametr přijímá klíč hradla *key*. Funkce *makeArrayToDelete()* vytvoří pole, které obsahuje ID zvoleného hradla a ID jeho potomků. Vytvořené pole, které obsahuje ID objektů k smazání, je procházeno a pokud se jedná o hradlo, zavolá se funkce pro smazání hradla *deleteGate* z *gateControlleru* a hradlo se z databáze a diagramu smaže. V případě události se zavolá funkce pro smazání události *deleteEvent* z *eventControlleru* a událost se smaže. Pokud zvolený objekt v diagramu je událost, zavolá se funkce pro smazání události *deleteEvent* z *eventControlleru* a smaže se událost, u které je ID události v databázi klíčem objektu *key* v označeném objektu. Po každém smazání objektu se zobrazí upozornění pro uživatele pomocí *alert()* s informací, který objekt byl právě smazán.

Pro případ vytvoření nového projektu lze celý strom z databáze a diagramu odstranit pomocí funkce *deleteObjects* z *viewControlleru* a celý předchozí systém se smaže z databáze i diagramu.

Zobrazení dat na základě parametrů

Stránky ve vytvořené aplikaci, zobrazující data na základě přijatých parametrů, přijímají data buď z Controlleru, nebo z URL. Pro zpracování parametrů z URL bylo použito API URLSearchParams. API bylo použito z důvodu získání parametrů z vytvořeného diagramu. Při označení daného hradla (události) v diagramu a zvolení z nabídky z možností přidání nebo editace hradla (události) se zobrazí stránka s požadovaným obsahem dynamicky vytvořená pomocí id objektu.

V přidání nového hradla (události) se ID zvoleného objektu vloží do pole pro předka objektu (*Parent Gate ID*). V případě editace se přidá do pole pro ID hradla (události).

Export diagramu

Vytvořený diagram lze exportovat do vektorového grafického formátu (SVG) nebo do rastrového grafického formátu (PNG). Pro export jsou použity implementované funkce v knihovně GoJS. Pro export do formátu SVG je na diagram použita funkce *makeSvg()*, kde v parametrech je zvoleno bílé pozadí diagramu. Uložený diagram v SVG formátu je následně převeden na datový textový řetězec String pomocí *XMLSerializer().serializeToString()*. Pro export do formátu PNG je použita funkce *makeImageData()*, která diagram rovnou převede na datový textový řetězec String. Datový textový řetězec je následně vložen do tzv. Blobu (nestrukturovaný soubor binárních dat). Po vytvoření nového elementu označeného tagem <a> na stránce, se do objektu URL této stránky vloží Blob, který je propojený odkazem v nově vytvořeném tagu. Po nasimulování kliku na tento tag, se diagram stáhne v požadovaném formátu do uložiště uživatele aplikace. [14]

Načítání dat ze souboru

Vstupní hodnoty událostí typu MTBF (střední doba mezi poruchami) a MTTR (střední doba do obnovy) lze načíst z textového souboru. Na stránce s formulářem pro přidání (editaci) základní události se pod položkami MTBF a MTTR nachází tag <input> s nastaveným typem na *file*, který zobrazí tlačítko pro načtení souboru z počítače. Po stisknutí se zobrazí dialogové okno s možností výběru souboru z uložiště počítače. Načíst lze textový soubor obsahující údaje o poruchách nebo obnovách v jednotkách hodin. Jednotlivé hodnoty jsou mezi sebou rozdělené středníkem. Po zvolení souboru se zavolá vytvořená funkce *getFileMTBF()* pro typ události MTBF nebo funkce *getFileMTTR()* pro typ události MTTR. Funkce *getFileMTBF()* a *getFileMTTR()* vytvoří střední hodnotu pomocí vytvořené funkce *calculateValue()* ze vstupních dat souboru. Funkce *calculateValue()* parsuje textový řetězec dat pomocí metody *split()*, která rozdělí textový řetězec na základě vstupujícího parametru na pole menších textových řetězců. V tomto případě je středník parametrem pro rozdělení. Po rozdělení se pokusí jednotlivé hodnoty převést pomocí funkce *parseFloat()* z textového řetězce na čísla, která jsou poté použita pro výpočet střední hodnoty. Funkce *calculateValue()* získá textový řetězec pomocí funkce *readFileContent()*, která načte obsah textového souboru pomocí rozhraní *FileReader()*.

Získaná střední hodnota se vloží buď do hodnoty MTBF nebo MTTR ve formuláři pro přidání (editaci) základní události. Pokud se zvolí soubor, který nemá správně definovaný formát dat, do hodnoty MTBF nebo MTTR se nevyplní žádný údaj.

Ukládání dat do souboru

Do souboru lze uložit výsledky po analýze systému v textovém formátu (txt). Do textového souboru se na každý řádek ukládají údaje o hradle nebo události ve formátu:

- Name – název hradla nebo události,
- Parent name – název předka (hradla),
- Time – čas, ve kterém byl počítán výsledek a jednotka času (uvedená v závorce),
- Result – výsledek hradla nebo události pro daný čas *Time*.

Jednotlivé hodnoty hradla nebo události jsou mezi sebou oddělené pomocí středníku. Vrcholová událost má nastavenou hodnotu pro *Parent name* na prázdný textový řetězec. Pro uložení v textovém souboru se uložená data v aplikaci vloží do Blobu a pomocí nově vytvořeného tagu <a> na stránce se stáhnou jako textový soubor do uložiště uživatele. Název souboru obsahuje informaci o typu analýzy, která byla použita při výpočtu.

Generování náhodného systému

Strom poruchových stavů lze náhodně vygenerovat pomocí možnosti *Generate Random Tree*, která se nachází v hlavní nabídce v sekci *Project*. Aplikace vygeneruje strom na základě přijatého počtu hradel, ze kterých se má systém sestavit. Náhodně vygenerovaný systém může mít až 30 hradel. Pro generování náhodných hodnot byla použita funkce *Math.random()*, která na základě přijatého rozmezí čísel vygeneruje číslo, které se nachází v zadaném rozmezí. Aplikace po výběru možnosti *Generate Random Tree* zobrazí dialogové okno pro zadání počtu hradel pomocí *window.prompt()*. Zadaná hodnota je nejdříve převedena na datový typ integer pomocí *parseInt()* pro případ, kdyby byla zadána jako desetinné číslo. Poté je zkontrolována pomocí regulárního výrazu, zda se jedná o číslo a ne textový řetězec. Na závěr je hodnota ověřena, zda byla zadána v rozmezí 1 až 30.

Generování náhodných hradel systému

Zadaný počet hradel poté vstupuje jako parametr do vytvořené funkce *startGeneratingTree()*, která náhodně vygeneruje systém. Funkce *startGeneratingTree()* vytváří 2 objekty typu JSON, ve kterých se nachází informace o hradlech a událostech ve vytvořeném diagramu. První objekt *objects_cal_array* obsahuje informace potřebné pro výpočet analýzy (včetně vstupních hodnot). Druhý objekt *jsonArrDiagram* obsahuje informace potřebné pro sestavení diagramu. Nejprve se do obou objektů vloží vrcholová událost, u které je nastaven typ hradla pomocí vytvořené funkce *generateGateType().* Funkce *generateGateType()* zvolí typ hradla na základě náhodně vygenerované hodnoty volby. Funkce umí vygenerovat hradlo typu AND (hodnota volby je 0) nebo hradlo typu OR (hodnota volby je 1). Po vložení vrcholové události do obou objektů

typu JSON se od zbývajícího počtu hradel odečte hodnota 1. Poté se dodefinují zbývající hradla systému, kde se pro již vytvořené hradlo vygeneruje hodnota, která určí počet jeho přímých potomků. Vygenerovaná hodnota se odečte od zbývajícího počtu hradel a nově vzniklá hradla jsou definována a vložena stejným způsobem do obou objektů typu JSON jako vrcholová událost. Algoritmus probíhá, dokud zbývající počet hradel není 0.

Generování náhodných základních událostí systému

Poté funkce *startGeneratingTree()* kontroluje hradla v systému, zda mají alespoň 2 přímé potomky. Pokud hradlo nemá alespoň 2 přímé potomky, jsou mu potomci doplněni náhodně vygenerovanými základními událostmi. Typ vstupních hodnot základní události se zvolí pomocí vytvořené funkce *generateEventType()* na základě náhodně vygenerované hodnoty volby. Pro základní událost lze generovat následující typy vstupních hodnot:

- constant – pravděpodobnost poruchy dána konstantní hodnotou (hodnota volby je 0),
- lambda – intenzita poruch (hodnota volby je 1).

Po nastavení typu vstupních hodnot základní události se nastaví vstupní hodnota pomocí vytvořené funkce *generateEventValue()*, která hodnotu náhodně vygeneruje. V posledním kroku nastavení události se pomocí vytvořené funkce *makeGeneratedEventChoice()* vytvoří textový řetězec, který se zobrazuje v zobrazeném diagramu u základní události a obsahuje informace o vstupních hodnotách. Vytvořený textový řetězec je hodnota klíče *choice* v objektu *jsonArrDiagram*.

Zobrazení náhodného systému

Po dokončení generování náhodného systému jsou oba objekty typu JSON (*objects_cal_array* a *jsonArrDiagram*) uloženy do *sessionStorage* a aplikace je přesměrována na hlavní pracovní prostředí, kde se náhodně vygenerovaný systém zobrazí stejným způsobem jako systém vytvořený manuálně uživatel. Náhodně vygenerovaný systém nenabízí možnosti pro přidání, editaci nebo smazání objektu (hradla nebo události), které se zobrazí v nabídce *ContextMenu*.

Uživatelský manuál

Na stránce *Help* se nachází vytvořený manuál ve formátu PDF k ovládání aplikace pro uživatele (viz Příloha 1). Soubor ve formátu PDF byl na stránku vložen pomocí HTML tagu *emded*, který umí zobrazit soubor, který není normálně podporován. Soubor PDF má na stránce povolené své možnosti ovládání (stažení, tisk, přiblížení, aj.). Na mobilních zařízeních se soubor ve formátu PDF zobrazuje deformovaně z důvodu zastaralosti HTML tagu *emded*. Z tohoto důvodu je na mobilních zařízeních místo zobrazeného souboru v PDF formátu pouze tlačítko pro stažení

manuálu do uživatelova mobilního zařízení. Odkaz na stránku *Help* se nachází na každé stránce aplikace, kde je potřeba vyplnit data ve formuláři (přidání hradla nebo události, nastavení analýzy). [19]

3.3 Výpočet analýzy systému

V této podkapitole je detailněji popsán proces výpočtu analýzy systému. Analýza vytvořeného systému probíhá v následujících krocích:

- kontrola vytvořených objektů v diagramu,
- nastavení analýzy systému,
- vytvoření časového vektoru,
- rozdělení hradel do vrstev,
- výpočet výstupních hodnot událostí,
- výpočet výstupních hodnot hradel,
- zobrazení výsledků.

V následující části je popsána kontrola vytvořeného diagramu před nastavením analýzy systému.

Kontrola vytvořených objektů v diagramu

Aplikace po vytvoření stromu poruchových stavů zkontroluje strom před vstupem na stránku s nastavením analýzy systému, jestli všechna hradla v diagramu mají správný počet vstupních událostí. Vytvořená funkce *checkTreeBeforeAnalysis()* kontroluje každé hradlo a pokud hradlo nemá dostatečný počet vstupních událostí (alespoň 2), aplikace se nepřesměruje na stránku s nastavením analýzy systému a zobrazí upozornění pomocí *alert()* s názvem hradla, které nemá dostatečný počet vstupních událostí. U hradla typu K/N se kontroluje, zda hradlo má alespoň K vstupních událostí. Pokud vytvořený strom má správně nastavené vstupní události pro všechna hradla, aplikace se přesměruje na stránku s nastavením analýzy.

Nastavení analýzy systému

Na stránce s nastavením analýzy systému se nachází formulář, který obsahuje následující položky potřebné k nastavení:

- Calculation Type – typ výstupní hodnoty událostí (viz Kapitola 1.7),
- Unit – jednotka času (sekundy, minuty, hodiny, dny nebo roky),
- Step – krok času, ve kterém má proběhnout výpočet (1, 5, 10, 50 nebo 100),
- Time – celkový čas, pro který má proběhnout analýza systému.

Položka formuláře *Calculation Type* má nastavenou výchozí hodnotu na *R(t)*. U položky *Unit* je nastavena výchozí hodnota na *hours* (hodiny) a u položky *Step* je nastaven výchozí krok výpočtu na 1. Aplikace u položky *Calculation Type* při výběru možnosti *A(t)* nebo *U(t)* zkontroluje vytvořený diagram, jestli mají všechny události nastavené správné parametry pro zvolený typ výstupních hodnot událostí pomocí vytvořené funkce *checkRepairableEventsForAnalysis()*. Funkce *checkRepairableEventsForAnalysis()* kontroluje každou událost v diagramu pomocí vytvořené funkce *checkEventsForMi()*, která jako vstupní parametr přijímá danou událost, u které ověřuje, zda má nastavené správné vstupní hodnoty pro zvolený typ výstupních hodnot událostí. Pro možnosti *A(t)* nebo *U(t)* lze počítat pouze s událostmi, které mají nastavené *Calculation Type* na jednu z následujících možností:

- Failure Rate (λ) and Repair Rate (μ) – intenzita poruch a intenzita oprav,
- MTBF and MTTR – střední doba mezi poruchami a střední doba do obnovy systému.

Pokud alespoň jedna událost v systému nemá nastavenou ani jednu z uvedených možností, funkce *checkRepairableEventsForAnalysis()* zobrazí pomocí *alert()* upozornění a položka formuláře *Calculation Type* je znovu nastavena na výchozí hodnotu *R(t)*.

Vytvoření časového vektoru

Po odeslání formuláře na stránce s nastavením analýzy systému proběhne samotný výpočet analýzy. Nejdřív aplikace pomocí vytvořené funkce *makeTimeVector()* vytvoří časový vektor *time*, na kterém probíhají jednotlivé dílčí výpočty analýzy. Funkce *makeTimeVector()* vytvoří časový vektor, ve kterém jsou uloženy původní časy v závislosti na zvoleném kroku v položce *Step* ve formuláři nastavení analýzy systému. Výsledný vektor slouží k zobrazení původních časů ve výsledcích analýzy a je uložen do uložiště *sessionStorage*, ze kterého je poté použit. Následně funkce z vektoru původních časů vytvoří nový vektor, který má jednotlivé hodnoty upravené v závislosti na zvolené jednotce času v položce *Unit* ve formuláři nastavení analýzy systému. Výpočet analýzy systému probíhá v jednotkách hodin. Pokud je zvolena jiná jednotka času, jsou jednotlivé časové hodnoty v dané jednotce převedeny do jednotek hodin. Výsledný vektor slouží k výpočtům výsledků analýzy.

Rozdělení hradel do vrstev

Aplikace po vytvoření časového vektoru rozdělí hradla v diagramu do vrstev pomocí funkce *get_layers_gates()*. Funkce *get_layers_gates()* zkoumá potomky každého hradla a rozděluje je do skupin (vrstev), které následně přidá do objektu typu JSON s názvem *groups_gates*.

Objekt *groups_gates* se skládá ze dvou klíčů:

- layer – číslo vrstvy,
- _ids – pole, které obsahuje ID hradel v dané vrstvě (*layer*).

První vrstva hradel je vrcholová událost systému. Druhá vrstva systému vznikne z hradel, jejichž přímým předkem je vrcholová událost. Další vrstva se naplní hradly, jejichž přímými předky jsou hradla v druhé vrstvě. Přidávání nových vrstev skončí, pokud se nenaleznou další potomci (hradla) systému ke zkoumání.

Výpočet výstupních hodnot událostí

Po rozdělení hradel do vrstev aplikace následně spočítá výstupní hodnoty událostí v diagramu pomocí funkce *get_event_results()*, do které vstupují jako parametry události systému, zvolený typ výstupních hodnot z nastavení analýzy a časový vektor, na kterém se hodnoty počítají. Ve funkci *get_event_results()* se pro každou událost spočítají zvolené výstupní hodnoty v závislosti na zadaných vstupních hodnotách události (viz Kapitola 1.6) v jednotlivých hodnotách časového vektoru.

Výsledné hodnoty událostí se uloží do objektu typu JSON s následujícími klíči:

- _id – ID objektu (v tomto případě události),
- parentID – ID předka objektu,
- values – pole výstupních hodnot události.

Objekt po naplnění hodnotami je dále použit pro dopočítání výstupních hodnot všech hradel v systému.

Výpočet výstupních hodnot hradel

Výstupní hodnoty jednotlivých hradel se vypočítají postupně od nejnižší vrstvy hradel (vrstva, která byla do objektu s vrstvami přidána jako poslední). Aplikace využívá hradel typu AND, OR a K/N. Výstupní hodnoty daného hradla se spočítají pomocí vzorců pro vyhodnocení kvantitativní analýzy (viz Kapitola 1.8). Pro výpočet výstupních hodnot hradel byly vytvořeny následující 3 funkce:

- and_gate – hradlo typu AND,
- or_gate – hradlo typu OR,
- kn_gate – hradlo typu K/N.

Do jednotlivých funkcí pro výpočet výstupních hodnot hradla vstupují jako parametr potomci daného hradla a typ výstupních hodnot (viz Kapitola 1.7), který byl zvolen ve formuláři nastavení analýzy. Do funkce *kn_gate* vstupuje navíc parametr K.

Pokud potomkem hradla je událost, má již vypočítané výstupní hodnoty v předchozí fázi výpočtu. Pokud potomkem hradla je hradlo, má také vypočítané výstupní hodnoty, jelikož se nachází v nižší vrstvě, která byla počítána dřív. Algoritmus výpočtu končí ve chvíli, kdy se spočítají výstupní hodnoty pro vrcholovou událost vytvořeného systému.

Zobrazení výsledků

Po dokončení analýzy systému se vypočítané výsledky uloží do uložiště *sessionStorage* a aplikace se přesměruje na stránku *Results*, ve které jsou získané výsledky analýzy zaznamenané v grafu (viz Příloha 2). Pro zobrazení výsledků analýzy pomocí grafu byla použita open source javascriptová knihovna Chart.js (https://www.chartjs.org/), která obsahuje nejrůznější grafové interpretace. K zobrazení výstupních hodnot vytvořeného systému je použit spojnicový graf. Osa x je pojmenována *Time* a obsahuje hodnoty z časového vektoru, na kterém proběhly výpočty. Osa y je pojmenována *Probability* a značí pravděpodobnost (vypočítanou výstupní hodnotu pro daný čas na ose x). [20]

Na stránce *Results* se nachází 2 vytvořené přepínače pro zobrazení konkrétních dat. Z uložiště *sessionStorage* se získají 2 objekty typu JSON. První objekt obsahuje veškeré údaje o hradlech a událostech ve vytvořeném systému. Druhý objekt obsahuje výstupní hodnoty (výsledky) jednotlivých hradel a událostí. První přepínač je naplněn hradly (událostmi) pomocí vytvořené funkce *fillResultSelect()* a obsahuje stručné informace o nich, které byly získány pomocí prvního objektu. Data jsou zde zobrazená v podobě:

- name – název hradla (události),
- type – typ hradla (události).

U hradla typu K/N se v možnosti hradla zobrazí také nastavený parametr K. Výchozí hodnota prvního přepínače je nastavena na vrcholovou událost. Při přepnutí na jiný objekt (hradlo nebo událost), aplikace pomocí funkce *result_select_change()* zjistí výstupní hodnoty zvoleného objektu a zobrazí nová data v grafu pomocí funkce *showGraph()*, do které jako parametr vstupuje vybraný objekt (hradlo nebo událost). Funkce *showGraph()* následně z přijatého objektu naplní graf hodnotami.

Druhým přepínačem na stránce *Results* lze přepínat mezi vypočítanou analýzou a analýzou, jejíž výstupní hodnoty jsou doplňkem k vypočítané analýze (viz Kapitola 1.7) a to pomocí funkce *values_type_change()*. Pro pravděpodobnost bezporuchového provozu R(t) lze tak ihned zobrazit výstupní hodnoty k pravděpodobnosti poruchy F(t). Pro funkci okamžité pohotovosti A(t) lze ihned zobrazit výstupní hodnoty k funkci okamžité nepohotovosti U(t). Pod druhým přepínačem se nachází tlačítko pro stažení grafu ve formátu JPEG.

Výstupní hodnoty všech objektů (hradel a událostí) se také zobrazí na hlavní uživatelské stránce ve vytvořeném diagramu. Pokud byla provedena analýza, každý objekt v diagramu má v zobrazených informacích zapsanou výstupní hodnotu vypočítanou v celkovém čase analýzy.

3.4 Zabezpečení aplikace

V této podkapitole je detailněji popsáno zabezpečení vytvořené webové aplikace. V následující části je popsáno řešení bezpečnosti přenosu přihlašovacích údajů uživatele mezi klientskou a serverovou částí aplikace.

Autentifikace uživatele

Aplikaci lze používat pouze pod vytvořeným uživatelským účtem. O uživateli se uchovává uživatelské jméno, email a heslo. Uživatelské rozhraní bylo vytvořeno jak z důvodu bezpečnosti, tak hlavně pro rozeznání uložených objektů v databázi, kde všechny objekty mají v parametrech uživatelské id, které je automaticky vygenerované díky MongoDB a je využíváno při veškerých operacích s objekty v databázi. Alternativně šlo použít rozeznávání objektů pomocí cookies.

Schéma uživatele má implementované funkce pro autentifikaci a uložení nového uživatele. Při posílání dat uživatele mezi serverovou a klientskou částí aplikace je zapotřebí heslo zasílat v hashovaném tvaru z bezpečnostních důvodů. Pro hashování uživatelských hesel byla použita knihovna bcrypt (https://www.npmjs.com/package/bcrypt), která obsahuje stejnojmennou hashovací funkci. Při hashování se používá tzv. sůl (náhodně vygenerovaný textový řetězec přidaný k heslu). Bcrypt má předdefinovanou funkci *genSaltSync()*, která jako parametr přijímá požadovanou délku náhodného textového řetězce. Ve vytvořené aplikaci má tento řetězec délku 10 znaků. V autentifikaci uživatele se vyhledává v databázi zadané uživatelské jméno. Pokud je nalezeno, tak se porovná zadané heslo s hashovaným heslem uloženém v databázi.

Pro zobrazení specifického obsahu přihlášeným uživatelům bylo potřeba nastavit session, která v aplikaci udržuje informace o přihlášených uživatelích. Pro vytvořenou aplikaci byl použit modul express-session (https://www.npmjs.com/package/express-session). Po úspěšném přihlášení se k id session přiradí id uživatele. Na základě tohoto přiřazení se pak zobrazí obsah uživatele, který je zobrazen po celou dobu, kdy je přihlášen. Po odhlášení uživatele se session „zničí“ a aplikace se přesměruje na přihlašovací obrazovku.

Bezpečnostní rizika

Vytvořená aplikace využívá databázi pro ukládání uživatelských dat. Databáze nese s sebou spoustu možných bezpečnostních rizik, které se musí před uvedením aplikace na trh vždy vyřešit. Špatně zabezpečená webová aplikace může způsobit ztrátu údajů uložených v databázi (např. informace o uživatelích, kreditních kartách, aj.). Dost útoků na webové aplikace probíhá

také pomocí vložení cizího objektu s nebezpečným kódem do nezabezpečeného textové výstupu, kdy se tento vložený objekt stane součástí výstupu aplikace a způsobí uživatelům ztráty dat nebo stažení nebezpečného softwaru do svého počítače (např. malware). V následujících části je popsána bezpečnostní hrozba NoSQL Injection.

NoSQL Injection

NoSQL Injection je bezpečnostní hrozba, která ohrožuje NoSQL databázi na základě špatně ošetřených vstupů aplikace. Příkazy v NoSQL databázi (např. MongoDB) se zapisují ve formátu JSON, přes který lze aplikaci napadnout, když nebudou dobře ošetřeny vstupy. Obrana proti NoSQL Injection spočívá ve filtrování vstupních dat před jejich odesláním na server. Na server se nesmí odeslat nekontrolovaná data od uživatele. [21]

Ve vytvořené aplikaci jsou veškeré vstupy ošetřeny před zasláním na server. NoSQL Injection by mohl být zacílen na stránky pro přidání hradla (události) nebo editaci hradla (události). Na těchto stránkách se pracuje s parametrem ID hradla (události), který se nachází v URL adrese a mohl by být přepsán na nebezpečný příkaz. V aplikaci byly vytvořeny funkce *parseAddURL()* a *parseEditURL()*, které parsují URL adresu a kontrolují vstupní hodnotu objektu, zda je ve správném formátu a neobsahuje žádné speciální znaky. Pokud obsahuje speciální znak, je stránka přesměrována na stránku zobrazující chybu 404 (stránka nenalezena).

Cross-site scripting

Cross-site scripting (XSS) je bezpečnostní hrozba, která ohrožuje webovou stránku tím, že ji vloží cizí JavaScript kód do neošetřeného vstupu. Tento kód se stane dočasně (non-persistent) nebo trvale (persistent) součástí této stránky a může způsobit únik informací uživatele nebo trvalé přesměrování na jinou stránku, na které se může nacházet jiná hrozba (např. malware). Obrana proti Cross-site scripting spočívá v ošetření vstupů, aby cizí kód nepřijímaly. [22]

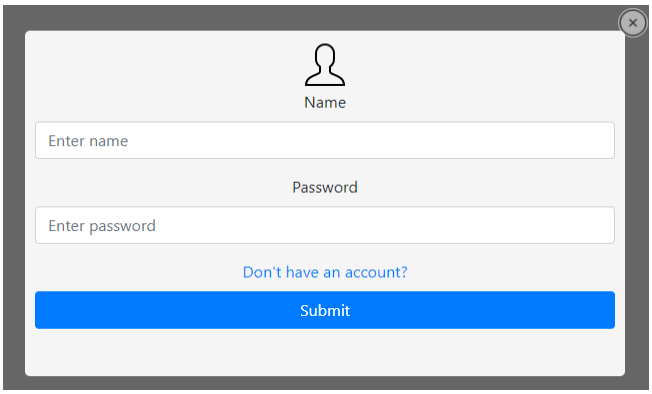
Ve vytvořené aplikaci by Cross-site scripting mohl napadnout formuláře pro vstup dat. Formuláře mají jednotlivé vstupy ošetřeny proti speciálním znakům. Pro tag *textarea*, která zobrazuje popis hradla nebo události, byla vytvořena speciální funkce *valid()*, která validuje každý znak a pokud se jedná o speciální znak, ihned ho smaže. To platí i pro případ vložení celého cizího kódu do tohoto tagu. Adresa URL je chráněna vytvořenými funkcemi *parseAddURL()* a *parseEditURL()*, které stejné jako v případě NoSQL Injection zkontrolují vstup a pokud obsahuje speciální znaky, tak ho nepřijme.

3.5 Ovládání aplikace

V této podkapitole je detailněji popsáno ovládání vytvořené aplikace z pohledu uživatele. V aplikaci se nachází uživatelský manuál, který je podrobnější a uživatel si ho může také stáhnout.

Rozhraní aplikace

Po spuštění webové aplikace se uživateli zobrazí přihlašovací stránka, kde se musí přihlásit svým účtem. Pokud nemá svůj účet vytvořený, může si jej vytvořit vyplněním registračního formuláře, kde vyplní jméno, emailovou adresu a heslo. Po registraci je uživatel přesměrován zpět k přihlášení.



Obrázek 19: Přihlášení uživatele do aplikace

Po úspěšném přihlášení se uživateli načte pracovní prostředí s hlavní nabídkou, kde se nachází ovládání aplikace v hlavní nabídce s následujícími možnostmi:

- Project – vytvoření nového projektu, generování náhodného systému, uložení výsledků a export diagramu,
- Analysis – spuštění analýzy a zobrazení výsledků,
- Help – manuál k ovládání aplikace,
- About – informace o aplikaci,
- Logout – odhlášení uživatele.

Vytvoření nového projektu

Pro vytvoření nového projektu uživatel zvolí z hlavní nabídky možnost *Project*, kde vybere položku *New*. Aplikace uživatele přesměruje na stránku k definování vrcholové události. Ve formuláři musí nastavit název vrcholové události (*Name*) a typ hradla (*Gate Type*). Na výběr má hradlo typu OR, AND nebo K/N. V případě zvolení hradla typu K/N musí ještě nastavit parametr K. Parametr N je zpočátku nastaven na výchozí hodnotu 0. Pokud událost potřebuje podrobnější popis, uživatel ho může zapsat v možnosti *Description*. Po vyplnění formuláře

se vrcholová událost přidá do uživatelského prostředí na vrchol. Přidané objekty lze v diagramu přesouvat a jednotlivé operace s objekty se nachází v nabídce, která se zobrazí po kliknutí na objekt pravým tlačítkem na myši. U mobilních zařízení se nabídka zobrazí pomocí delšího kliku na objekt. U hradel nabídka obsahuje možnosti pro přidání dalšího hradla nebo události, smazání a editaci hradla. U události obsahuje pouze možnosti pro smazání a editaci. U nově přidaného hradla uživatel může vybrat z hradel typů AND, OR nebo K/N.

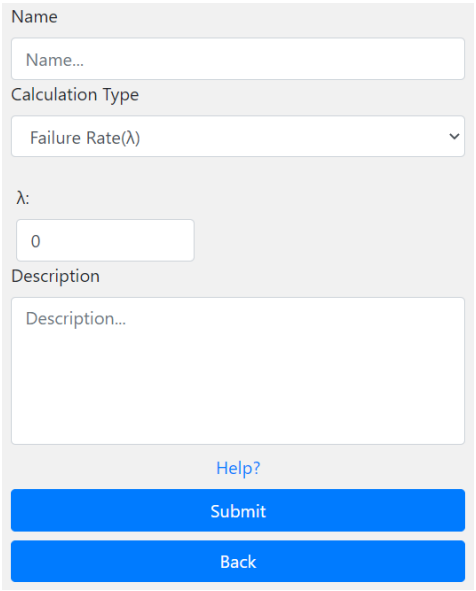
U nově přidané události se musí vyplnit název (*Name*) a zvolit typ výpočtu (*Calculation Type*), kde jsou na výběr následující typy pro neobnovitelné systémy:

- Constant – pravděpodobnost poruchy určená konstantní hodnotou,
- Failure Rate (λ) – intenzita poruch,
- MTBF – střední doba mezi poruchami,

Pro obnovitelné systémy lze zvolit z následujících možností:

- Failure Rate (λ) and Repair Rate (μ) – intenzita poruch a intenzita oprav,
- MTBF and MTTR – střední doba mezi poruchami a střední doba do obnovy (opravy).

U typů MTBF a MTTR může uživatel hodnotu zadat přímo nebo ji může nechat aplikací vypočítat ze zvoleného souboru, ve kterém jsou zaznamenány buď časy mezi poruchami pro MTBF nebo časy do obnovy pro MTTR. Uživatel může událost detailněji popsat v možnosti *Description*.



Obrázek 20: Formulář pro přidání nové události

Pokud uživatel potřebuje upravit údaje hradla nebo události, vybere z nabídky objektu možnost *Edit*, která ho přesměruje k formuláři pro editaci hradla nebo události. Pro smazání hradla nebo události se vybere z nabídky možnost *Delete*. Uživatel je o každém smazaném objektu

informován, že byl smazán úspěšně. Pro lepší orientaci se v aplikaci nachází manuál pod kolonkou *Help*, kde uživatel může vyčíst veškeré ovládání aplikace v zde zobrazeným souboru ve formátu PDF. Manuál si také může stáhnout a otevřít z počítače. U mobilních zařízení je manuál k dispozici pouze ke stažení.

Po vytvoření stromu poruchových stavů se uživatel dostane ke spuštění analýzy pomocí hlavní nabídky, kde zvolí položku *Analysis* a následně vybere možnost *Start*. V položce *Analysis* se nachází také možnost *Results* pro zobrazení výsledků, do které má uživatel možnost vstoupit až po dokončení analýzy systému. Aplikace nejdříve před nastavením analýzy zkontroluje vytvořený diagram, zda hradla mají dostatečný počet vstupů. Pokud nemají, zobrazí se uživateli upozornění se zprávou, jaké hradlo nemá správný počet vstupů. Uživatel musí diagram zkontrolovat a po doplnění vstupů může do sekce s nastavením analýzy systému.

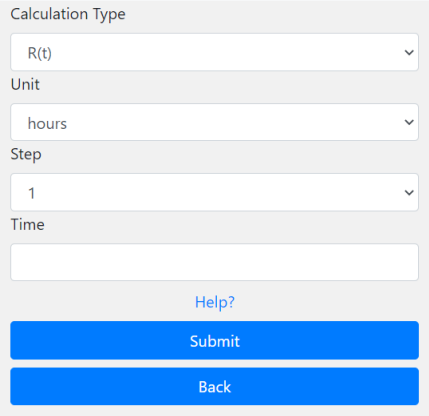
Nastavení analýzy systému

V nastavení analýzy uživatel musí nastavit typ analýzy (*Calculation Type*), jednotku času (*Unit*), krok analýzy (*Step*) a definovat čas (*Time*), pro který se má analýza systému uskutečnit. Uživatel má na výběr z následujících typů analýzy:

- R(t) – pravděpodobnost bezporuchového provozu,
- F(t) – pravděpodobnost poruchy,
- A(t) – funkce okamžité pohotovosti,
- U(t) – funkce okamžité nepohotovosti.

Funkce okamžité pohotovosti a nepohotovosti systému může zvolit pouze tehdy, pokud mají všechny vstupní události nastavený typ výpočtu *Failure Rate (λ) and Repair Rate (μ)* nebo typ výpočtu *MTBF and MTTR*. Pokud nemá nějaká událost nastavený uvedené typy výpočtu, nelze danou analýzu systému uskutečnit.

Následně může změnit jednotky času v položce *Unit*, ve kterých se analýza systému bude počítat. Výchozí hodnota je nastavena na hodiny, ale uživatel ji může změnit na sekundy, minuty, dny nebo roky. V položce *Step* lze nastavit časové kroky, ve kterých mají probíhat výpočty hodnot analýzy. Výchozí hodnota je nastavena na 1. Mezi další možnosti nastavení kroku patří 5, 10, 50 a 100. Pokud uživatel nastaví krok větší než je nastaven čas v položce *Time*, hodnota kroku se nastaví na 1. V poslední položce *Time* se nastavuje čas, do kterého má probíhat zvolený typ analýzy systému.



Obrázek 21: Formulář pro nastavení analýzy

Zobrazení výsledků analýzy

Po dokončení analýzy je uživatel přesměrován na stránku *Results*, ve které jsou výsledky analýzy zobrazené pomocí grafu (viz Příloha 2). Na této stránce se nachází 2 přepínače. První slouží k výběru objektu (hradla nebo události), kterému se mají zobrazit výsledky v grafu. Výchozí výběr přepínače je nastaven na vrcholovou událost ve stromu. Druhý přepínač slouží k výběru analýzy. Výchozí výběr je nastaven na provedenou analýzu systému. Pro snažší ovládání aplikace lze přepínat mezi analýzami, jejichž výsledky jsou si navzájem doplňkem. U pravděpodobnosti bezporuchového provozu lze tak zobrazit výsledky pro pravděpodobnost poruchy. U výsledků funkce okamžité pohotovosti lze přepnout na výsledky funkce okamžité nepohotovosti. Graf s výsledky analýzy si může uživatel stáhnout pomocí tlačítka, které se nachází pod druhým přepínačem.

Výsledky analýzy se zobrazí i v diagramu na hlavní pracovní ploše uživatele. Hradla a události zobrazí výsledek v čase, který uživatel nastavil při nastavení analýzy v položce *Time.* Celý průběh analýzy všech hradel a událostí může uživatel stáhnout v podobě textového souboru z hlavní nabídky výběrem Project a možnosti Save Results.

Generování náhodného systému

Strom poruchových stavů může uživatel také vytvořit pomocí možnosti *Generate Random Tree*, která se nachází v hlavní nabídce v položce *Project.* Možnost *Generate Random Tree* vytvoří náhodný systém na základě počtu hradel, který je zadán ve vyskakujícím dialogovém okně. Aplikace umí náhodně vygenerovat systém až o 30 hradlech. Po zadání počtu hradel, které má systém obsahovat, se vytvoří systém s náhodně nastavenými hradly a událostmi. Takto vytvořený systém nenabízí možnosti pro přidání, editaci či smazání objektu (hradla nebo události), které se zobrazují pomocí kliku pravého tlačítka myši na objekt.

3.6 Porovnání vytvořené aplikace s komerčními aplikacemi

Výhody a nevýhody jednotlivých aplikací byly probrány v předchozí části práce. V této podkapitole se nachází srovnání vlastností vytvořené aplikace s komerčními softwary, které byly podrobně popsány v kapitole 2. Softwary rozebrané v rámci této bakalářské práce se dělí na webové a desktopové. Mezi webové softwary patří:

- Fault Tree Analyser,
- vlastní aplikace.

Mezi desktopové softwary patří:

- ITEM ToolKit,
- Isograph,
- ReliaSoft,
- RAM Commander.

Obsáhlost softwarů

Komerční softwary jsou obsáhlé aplikace, které nabízejí více typů analýz pro zkoumání systémů. Nejobsáhlejším softwarem byla aplikace ITEM ToolKit, která obsahovala mnoho doplňkových balíčků pro nejrůznější typy analýz. Nejméně obsáhlou aplikací byla vytvořená aplikace v rámci této práce. Nabízí pouze metodu FTA (analýza stromu poruchových stavů).

Uživatelské prostředí

Nejvíce uživatelsky přívětivou aplikací je aplikace vytvořená v rámci této práce, která nabízí jednoduché prostředí, v kterém se lze dobře vyznat. Obsahuje i manuál, ve kterém jsou veškeré úkony aplikace popsány. Nejméně uživatelsky přívětivou aplikací je aplikace ITEM ToolKit, která neobsahuje jednoduché uživatelské prostředí. Člověk, který není příliš znalý v oboru, se moc nevyzná v práci s touto aplikací, která je určená spíše pro znalé v oboru.

Počet objektů v diagramu

Komerční softwary byly zkoumány pouze v demo verzích, které byly po určitou zdarma a měly určitá omezení. Omezení se hlavně týkala počtu vložených objektů. Nejvíce objektů (hradel a událostí) lze vložit ve vytvořené aplikaci v rámci této práce. Pro testovací účely byl vygenerován i systém o 65 hradlech. Naopak nejméně objektů šlo vložit v aplikaci RAM Commander, která pracovala maximálně s 15 hradly.

4 Ověření správnosti výpočtu výsledků

V poslední části bakalářské práce je popsáno ověření správnosti výpočtu analýzy na základě testovaného příkladu. K ověření správnosti výpočtu analýzy ve vytvořené aplikaci byl napsán skript pomocí programovacího jazyka MATLAB a výsledné hodnoty analýzy byly porovnány. V následující podkapitole je popsán systém, který byl vytvořen k testování.

4.1 Testovaný příklad

Ve vytvořené aplikaci byl vytvořen systém (viz Příloha 3), který se skládá z následujících objektů:

- 15 hradel (typu AND nebo OR),
- 17 základních událostí.

Vstupní hodnoty základních událostí byly nastaveny na pravděpodobnosti poruch, které jsou dány konstantní hodnotou nebo na intenzity poruch. Hodnoty pravděpodobností a intenzit poruch byly zvoleny náhodně. V nastavení analýzy byly vybrány následující parametry:

- typ analýzy – pravděpodobnost bezporuchového provozu R(t),
- jednotka času – hodiny,
- krok výpočtu – 5,
- počet hodin – 100.

Po výpočtu analýzy systému byly výsledné hodnoty analýzy porovnány s hodnotami získanými pomocí vytvořeného skriptu v programovacím jazyce MATLAB.

4.2 Ověření pomocí programovacího jazyka MATLAB

Vytvořený skript v programovacím jazyce MATLAB má vstupní hodnoty testovaného systému uložené v polích. Ve vytvořeném skriptu byly zapsány následující vstupní hodnoty z testovaného systému:

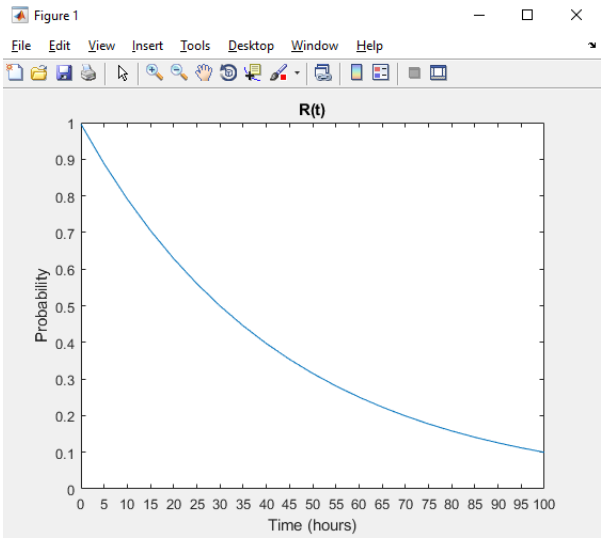
- time – časový vektor, na kterém proběhla analýzy,
- gate_types – pole, které obsahuje typy použitých hradel (OR nebo AND),
- gate_parents – pole, které obsahuje ID předků hradel,
- event_types – pole, které obsahuje typy použitých základních událostí (pravděpodobnosti poruch nebo intenzity poruch),
- event_parents – pole, které obsahuje ID předků základních událostí,
- event_values – pole, které obsahuje hodnoty pravděpodobností poruch nebo intenzit poruch.

Výpočet testovaného příkladu

Skript nejdříve spočítá výstupní hodnoty všech základních událostí a uloží je do pole s názvem *events_results*. Výstupní hodnoty hradel se ukládají do pole s názvem *gates_results*, které bylo inicializováno na začátku jako pole obsahující pouze hodnoty 0. Výstupní hodnoty hradla se do pole *gates_results* přidávají od konce (od posledního hradla uloženého v systému). Skript ověří, zda hradlo má nějaké přímé potomky ve vytvořených polích *gate_parents* a *event_parents*. Pokud je přímý potomek hradla událost, přidají se výstupní hodnoty události z pole *events_results* do pole *results_to_calculate.* Pokud přímým potomkem je hradlo, přidají se výstupní hodnoty hradla z pole *gates_results*, kde už z předchozích kroků má potomek vypočítané výstupní hodnoty, které se přidají do pole *results_to_calculate*. Poté se výstupní hodnoty hradla vypočítají podle zvoleného typu hradla (OR nebo AND) z výstupních hodnot, které jsou uložené v poli *results_to_calculate*. Nově vzniklé výstupní hodnoty hradla se uloží do pole *gates_results*. Algoritmus probíhá, dokud nebudou vypočítány výstupní hodnoty všech hradel v systému.

Porovnání výstupních hodnot testovaného příkladu

Po výpočtu všech výstupních hodnot hradel a událostí v testovaném systému se zobrazí výstupní hodnoty vrcholové události v konzoli a v grafu (viz Obrázek 22). Graf se zobrazí pomocí funkce *plot()*. Na ose x je nanesen časový vektor *time.* Na ose y jsou nanesené jednotlivé výstupní hodnoty (pravděpodobnosti bezporuchového provozu) vrcholové události systému v závislosti na čase. Výstupní hodnoty byly získané z pole *gates_results*, kde se vrcholová událost systému nachází na indexu 1.



Obrázek 22: Zobrazení grafu s výsledky vrcholové události v jazyce MATLAB

Výstupní hodnoty vrcholové události byly porovnány s hodnotami získanými pomocí aplikace. V následující tabulce (viz Tabulka 1) jsou zapsány výstupní hodnoty z vytvořené aplikace a ze skriptu napsaného v programovacím jazyce MATLAB.

Tabulka 1: Výstupní hodnoty vrcholové události z aplikace a ze skriptu

Čas (hodiny)	Výstupní hodnoty	
	Aplikace	MATLAB
0	0.9958016110174618	0.9958
5	0.887618640962749	0.8876
10	0.7911885854665591	0.7912
15	0.7052346150625227	0.7052
20	0.6286186144751684	0.6286
25	0.5603261127462891	0.5603
30	0.49945285055135513	0.4995
35	0.4451928068414211	0.4452
40	0.39682752626924866	0.3968
45	0.3537166060823361	0.3537
50	0.31528921651830627	0.3153
55	0.28103654242292103	0.2810
60	0.2505050460090959	0.2505
65	0.22329046154816698	0.2233
70	0.19903244247632756	0.1990
75	0.17740979003789448	0.1774
80	0.15813620028729838	0.1581
85	0.1409564731353685	0.1410
90	0.12564313324348636	0.1256
95	0.11199341802252494	0.1120
100	0.09982659285438075	0.0998

Vytvořený skript počítal jednotlivé výstupní hodnoty systému se zaokrouhlením na 4 desetinná místa. Aplikace jednotlivé hodnoty pro výpočet výstupních hodnot nezaokrouhlovala. Po odečtení hodnot z aplikace a ze skriptu vzniká tedy tzv. zaokrouhlovací chyba, která ale nemá v takto definovaném systému zásadní roli. Pokud se výstupní hodnoty testovaného systému z aplikace zaokrouhlí na 4 desetinná místa, vyjdou identické hodnoty k získaným hodnotám ze skriptu. Případný problém se zaokrouhlováním výstupních hodnot by mohl nastat s komplexnějšími systémy, které obsahují mnohem více hradel a událostí. Zde by mohla větší zaokrouhlovací chyba znamenat odlišné výsledky.

Závěr

Cílem bakalářské práce bylo seznámení se spolehlivostní metodou analýzy stromu poruchových stavů (FTA), její důkladné prostudování a vytvoření vlastní aplikace. K nastudování metody FTA existuje spousta materiálů a veškeré informace, které se vázaly k metodě, se daly najít bez problému v literatuře. Informace o metodě byly převážně vzaty z technických norem.

V druhé části práce byla zpracována rešerše existujících komerčních softwarů, kdy ke zpracování byly použity převážně demo verze softwarů zaslané přímo společností. U softwaru ReliaSoft byl společností HBM Prenscia omylem zaslán klíč k jiné aplikaci. K nedorozumění došlo v komunikaci s podporou, která ale ihned poslala klíč správný a telefonicky se omluvila. Rešerše softwarů byla hlavně zaměřena na práci se sestavováním stromu poruchových stavů. Každý software k sestavování přistupoval odlišným způsobem a to pomohlo k naleznutí optimálního způsobu sestavení systému, který byl použit při tvorbě stromu poruchových stavů ve vytvořené aplikaci.

V třetí části této práce byla popsána tvorba vlastního softwaru počítajícího metodu FTA. K vytvoření softwaru pomohla knihovna GoJS, která obsahuje nadefinovanou strukturu zobrazení diagramu pro metodu FTA. Software počítá se základními hradly (OR, AND, K/N), ale může se snadno rozšířit o další specifická hradla. To samé platí pro události. V aplikaci byl vytvořen generátor náhodných událostí, který usnadnil následné testování systému s větším počtem objektů.

V závěrečné části této práce byla ověřena správnost výpočtu výstupních hodnot systému ve vytvořené aplikaci na základě testovaného příkladu. Výstupní hodnoty z aplikace byly porovnány s hodnotami získanými z vytvořeného skriptu v programovacím jazyce MATLAB. Skript jednotlivé výsledky zaokrouhloval na 4 desetinná místa. V aplikaci se jednotlivé výpočty zaokrouhlují až při výpisu do textových řetězců. Na testovaném systému se zaokrouhlovací chyba nijak neprojevila a při zaokrouhlení výsledků v aplikaci na 4 desetinná místa vyjdou identické hodnoty jako ze skriptu.

Seznam použité literatury

[1] ČSN EN 61025 (010676) *Analýza stromu poruchových stavů (FTA)*. Praha: Česká technická norma, 2007.

[2] FTA (Fault Tree Analysis) - Analýza stromu poruchových stavů. ManagementMedia [online]. [cit. 2019-10-04]. Dostupné z: https://managementmania.com/cs/fault-tree-analysis

[3] FUCHS, Pavel, Vališ DAVID, Chudoba JOSEF, Kamenický JAN a Zajíček JAROSLAV. *Řízení spolehlivosti* [online]. Liberec: Technická univerzita v Liberci, 2006 [cit. 2019-10-04].

[4] Weibull.com - *Fault Tree Analysis* [online]. [cit. 2019-10-04]. Dostupné z: https://www.weibull.com/basics/fault-tree/index.htm

[5] Fault Tree Handbook. In: . Washington, D.C., 1981. Dostupné také z: http://www.nrc.gov/docs/ML1007/ML100780465.pdf

[6] ČSN IEC 50 (191)(010102). *Medzinárodný elektrotechnický slovník, kapitola 191: Poľahlivosť a akosť služieb*. 1993.

[7] *Fault Tree Analyser* [online]. [cit. 2019-10-20]. Dostupné z: https://www.fault-tree-analysis-software.com/ald-reliability-safety-engineering

[8] *ITEM ToolKit* [online]. [cit. 2019-10-22]. Dostupné z: https://www.itemsoft.com/about_us.html

[9] *Isograph* [online]. [cit. 2019-11-10]. Dostupné z: https://www.isograph.com/

[10] *ReliaSoft* [online]. [cit. 2019-11-22]. Dostupné z: https://www.reliasoft.com/about

[11] *RAM Commander* [online]. [cit. 2019-12-15]. Dostupné z: https://aldservice.com/RAMS-Reliability-Availability-Maintainability-and-Safety-Software.html

[12] *MVC: Model, View, Controller*. *Codecademy* [online]. [cit. 2020-01-15]. Dostupné z: https://www.codecademy.com/articles/mvc

[13] *About Heroku. Heroku* [online]. [cit. 2020-02-23]. Dostupné z: https://www.heroku.com/about

[14] *GoJS* [online]. [cit. 2020-02-24]. Dostupné z: https://gojs.net/latest/index.html

[15] *Treant.js* [online]. [cit. 2020-02-24]. Dostupné z: https://fperucic.github.io/treant-js/

[16] *Mongoose* [online]. [cit. 2020-03-13]. Dostupné z: https://mongoosejs.com/

[17] JSON - Úvod do JSON. *JSON* [online]. [cit. 2020-03-23]. Dostupné z: https://www.json.org/json-cz.html

[18] *Window.sessionStorage*. MDN Web Docs [online]. [cit. 2020-04-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage

[19] *Objekty v HTML*. Jak psát web [online]. [cit. 2020-04-10]. Dostupné z: https://www.jakpsatweb.cz/html/objekty.html

[20] *Chart.js documentation*. Chart.js [online]. [cit. 2020-04-12]. Dostupné z: https://www.chartjs.org/docs/latest/

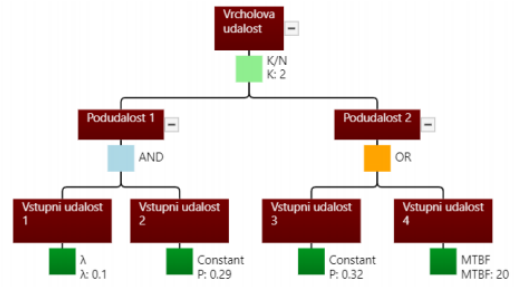
[21] *What is NoSQL injection?* Infosec Resources [online]. [cit. 2020-04-24]. Dostupné z: https://resources.infosecinstitute.com/what-is-nosql-injection/#gref

[22] *Cross-site Scripting (XSS).* Acunetix [online]. [cit. 2020-05-04]. Dostupné z: https://www.acunetix.com/websitesecurity/cross-site-scripting/

Přílohy

Manuál k aplikaci FTA

Aplikace FTA se zabývá stejnojmenným typem analýzy FTA (Fault Tree Analysis – v češtině Analýza stromu poruchových stavů). Jde vesměs o zkoumání vrcholové události, která je rozčleněná do menších podudálostí (viz. Obrázek 1), které tuto událost s určitou pravděpodobností v čase vyvolají.



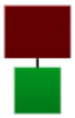
Obrázek 1: Ukázka diagramu FTA

Seznámení s objekty

V aplikaci se využívá pouze objektů typu událost a hradlo.

Události

Události jsou značky, které značí výskyt určitého děje. Níže jsou popsány vstupní a výstupní hodnoty události.



Vstupní hodnoty události

V aplikaci je na výběr z 5 typů vstupních hodnot pro událost popsaných níže.

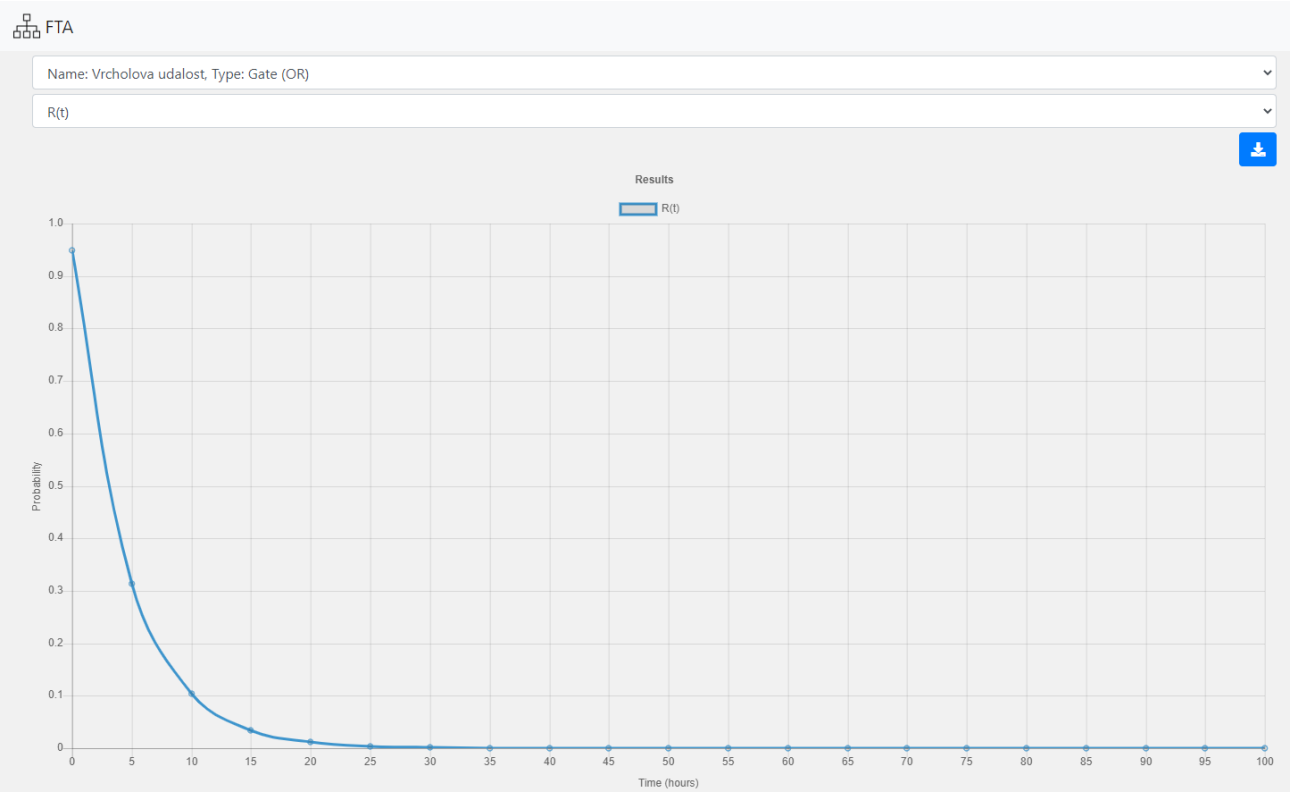
Constant

Pravděpodobnost poruchy nastavená konstantní hodnotou. Na celém časovém intervalu zůstává konstantní (stejná).

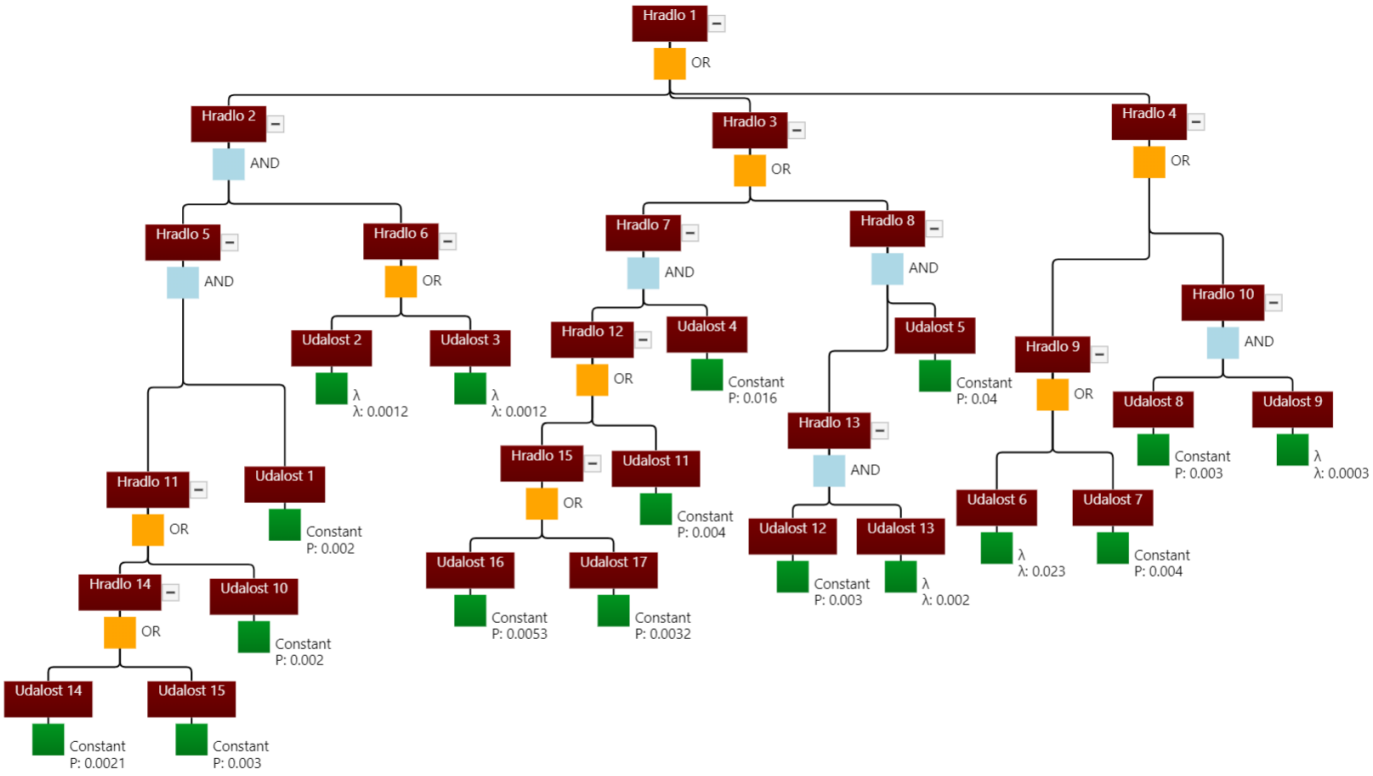
MTBF

Mean Time Between Failures (v češtině) střední doba mezi poruchami) je očekávána doba provozu mezi poruchami. Z MTBF se získává intenzita poruch. Aplikace přijímá hodnotu MTBF v hodinách

Příloha 1: Manuál k aplikaci dostupný na stránce Help



Příloha 2: Zobrazení výsledků na stránce Results



Příloha 3: Testovaný systém pro ověření správnosti výpočtu analýzy