TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Aplikace řešící spolehlivost systémů metodou stromu poruchových stavů (FTA)

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Jakub Nožička**
Vedoucí práce: Ing. Josef Chudoba, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jakub Nožička**
Osobní číslo:	**M13000198**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace řešící spolehlivost systémů metodou stromu poruchových stavů (FTA)**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Prostudujte spolehlivostní metodu FTA a způsoby zálohování systému.
2. Definujte vlastnosti základních komponent metody FTA a implementace do softwarového nástroje.
3. Vytvořte aplikaci a proveďte výpočty základních spolehlivostních parametrů.
4. Příklad a testování použité aplikace, dokumentace.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **50 - 60 stran**

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

Seznam odborné literatury:

[1] Fuchs P., Vališ D. Chudoba J., Kamenický J., Zajíček J., Řízení spolehlivosti, učební text, Technická univerzita v Liberci 2006
[2] ČSN EN 61025 (010676) Analýza stromu poruchových stavů (FTA), Česká technická norma, 2007
[3] Pecinovský R., Návrhové vzory, Computer press, Praha 2007, ISBN 978-80-2511-582-4
[4] Pecinovský R., OOP naučte se myslet a programovat objektově, Computer press, Praha 2010, ISBN 978-80-2512-126-9
[5] Knuth D. E., Umění programování, Computer press, Praha 2010, ISBN 978-80-2512-898-5

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

Datum zadání diplomové práce: **20. října 2015**
Termín odevzdání diplomové práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tom-to případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15. 5. 2017

Podpis:

Poděkování

Rád bych poděkoval všem, kteří se podíleli na vzniku této diplomové práce. Děkuji svému vedoucímu Ing. Josefu Chudobovi, Ph.D. za pomoc, rady a příkladné vedení této diplomové práce. Nakonec bych rád poděkoval svým blízkým za podporu, díky které jsem mohl práci dokončit.

Abstrakt

Tato práce se zabývá vytvoření SW aplikující spolehlivostní metodu FTA (Fault Tree Analysis). Je zde uveden princip metody, způsob a implementace výpočtu. Cílem práce je vytvoření softwaru pro výpočet pravděpodobnosti bezporuchového provozu systému a pravděpodobnosti poruchy za využití metody FTA.

Informace o metodě FTA byly čerpány z technických norem a dalších písemných i elektronických zdrojů, které se metodou zabývají. Dalšími zdroji informací byly komerční softwary, které se zabývají řešením spolehlivosti systémů. Jedním z cílů práce bylo testování těchto softwarů a jejich porovnání s vyvinutým SW.

Pro tvorbu výsledného programu byl využit programovací jazyk C#. Program poskytuje výpočet pravděpodobnosti bezporuchového systému, výpočet pravděpodobnosti poruchy a výpočty okamžité funkce pohotovosti a nepohotovosti. Dále poskytuje grafické zobrazení výsledků pro lepší přehlednost.

Klíčová slova: FTA, software, spolehlivost, bezporuchovost, pravděpodobnost

Abstract

This project deals with reliability using FTA method (Fault Tree Analysis). How method works and what it is purpose. Purpose work is creating software to calculate the probability of reliability of the system and the probability of failure using FTA method.

Information about the method FTA was gathering from technical standards and other written and electronic resources. Other resources of information were commercial software which deals to solution of system reliability. One of purpose project was to test this software.

For create of the final software was used programming language C#. Software provides the calculation of the probability of reliability system, calculation of the probability of failure and calculation of immediate availability and unavailability functions. Software also provides a graphical display of results.

Key words: FTA, software, dependability, reliability, probability

Obsah

Seznam obrázků

Seznam zkratek a použité značení

$\boldsymbol{A(t)}$	[1]	funkce okamžité pohotovosti
ETA		analýza stromu událostí
FMEA		analýza druhů poruchových stavů a jejich důsledků
FMECA		analýza druhů, důsledků a kritičnosti poruchových stavů
FTA		analýza stromu poruchových stavů
RBD		blokový diagram bezporuchovosti
$\boldsymbol{F(t)}$	[1]	pravděpodobnost poruchy
$\boldsymbol{\lambda(t)}$	$[hod^{-1}]$	intenzita poruch
$\boldsymbol{MTTF}$	$[hod]$	střední doba do poruchy
$\boldsymbol{MTTR}$	$[hod]$	střední doba do obnovy
$\boldsymbol{\mu(t)}$	$[hod^{-1}]$	intenzita oprav
$\boldsymbol{R(t)}$	[1]	pravděpodobnost bezporuchového provozu
$\boldsymbol{t}$	$[s]$	čas
$\boldsymbol{U(t)}$	[1]	funkce okamžité nepohotovosti

Úvod

S termínem spolehlivost se setkáváme téměř na každém kroku. Obvykle jde o spolehlivost nějakého přístroje nebo zařízení. Obyčejný uživatel hodnotí spolehlivost podle toho, jak je s přístrojem nebo zařízením spokojen. Při návrhu systému, přístroje nebo zařízení se spolehlivost řeší z jiného úhlu pohledu, celý budoucí projekt závisí na tom, zda systém nebo přístroj bude spolehlivě pracovat či ne. Např. při konstrukci součástek nebo systémů pro jadernou elektrárnu nebo pro kosmický program je spolehlivost jednou z nejdůležitějších vlastností systémů. Pro výpočet spolehlivosti existuje několik programů od společností, které se spolehlivostí zabývají.

Diplomová práce se zabývá jednou z mnoha metod pro výpočet spolehlivosti, a to metodou FTA (metoda stromu poruchových stavů) [1]. První část práce je věnována samotné metodě FTA. Popisuje se zde princip metody, dále jaké metoda používá vstupy a jaké jsou její výstupy. Jsou zde popsány i základní grafické symboly, které metoda využívá při tvorbě stromu poruch.

V další části práce (kap. 2) se nachází popis několika testovaných softwarů od společností, které se zabývají problematikou spolehlivosti systémů. U každého programu je popsán princip vkládání nových objektů, zadávání vstupních hodnot a zobrazení výstupních hodnot. Dále je u každého testovaného softwaru zhodnocení.

V závěrečné části (kap. 3) pak nalezneme popis vlastního vytvořeného programu pro výpočet spolehlivosti pomocí metody FTA. Je zde popsán algoritmus samotného výpočtu; jednotlivé objekty, které program využívá pro tvorbu stromu poruch nebo jaké formy výstupu program nabízí. Nachází se zde dokumentace k programu, kde jsou popsány jednotlivé části programu a nakonec závěrečná část obsahuje testovací příklad programu a jeho výsledky.

1 FTA - Fault Tree Analysis

1.1 Popis metody FTA

Analýza stromu poruchových stavů je jedna z metod pro výpočet spolehlivosti a bezpečnostní analýzy. Mezi další metody v oblasti řízení a spolehlivosti patří např. metody:

- ETA (Event Tree Analysis)
- FMEA (Failure Mode and Effect Analysis).
- RBD (Reliability Block Diagram)
- Markovská analýza

Většinou se metoda využívá jako preventivní, často také po poruše. Jedná se o deduktivní metodu. Model se vytváří tzv. shora dolů, to představuje, že definovaná vrcholová událost se větví na jednotlivé události a příčiny, které mohou tuto vrcholovou událost způsobit. Analýzy bezporuchovosti systému jsou zaměřeny na přesném určení příčin, nebo častěji kombinaci jednotlivých příčin, které mohou vést k námi definované vrcholové události.

Metodu FTA lze provádět nezávisle na jiných analýzách bezporuchovosti nebo společně s nimi. Metoda FTA je vhodná zvláště k analýze systémů, které se skládají z několika funkčně souvisejících nebo závislých podsystémů. Obecně se metoda FTA používá při projektování dopravních systémů, jaderných elektráren, chemických a jiných průmyslových procesů, zdravotních systémů, počítačových systémů a mnoho dalších. Dále se metoda využívá u systémů skládajících se z rozmanitých typů součástí (mechanických, elektronických nebo softwarových) a jejich vzájemných interakcí, které nelze snadno modelovat jinými technikami.

Metoda FTA má rovněž využití jako nástroj pro stanovení příslušné logické kombinace jednotlivých událostí vedoucích k vrcholové události. Dále ke zkoumání systému v průběhu jeho vývoje a předvídání, zabránění vzniku nebo zmírnění následků příčin vrcholové události. Využívá se k analýze systému, určení jeho bezporuchovosti, k hledání a identifikaci hlavních „přispěvatelů“ k jeho poruchovosti a jako pomocná metoda při pravděpodobnostním posuzování rizika. [1] [8]

1.2 Cíle metody FTA

Mezi hlavní cíl metody patří identifikovat příčiny nebo kombinaci příčin, které mohou zapříčinit vrcholovou událost. [1] Mezi další cíle metody FTA patří:

- prokazování, že předpoklady, které byly učiněny v jiných analýzách (FMEA nebo Markovova analýza) jsou správné,
- vyhledat událost nebo kombinace událostí, které s největší pravděpodobností stojí za vznikem vrcholové události,
- vypočítat pravděpodobnosti výskytu vrcholové událostí.

1.3 Fault Tree Diagram

Fault Tree Diagram (strom poruchových stavů) je grafické znázornění daného problému. Tento diagram tvoří strom, kde jako vrcholová událost (kořen) je námi hledaný kritický problém. Tato událost se postupně větví na menší události, které mohou tuto vrcholovou událost ovlivnit, nebo ji přímo způsobit. Strom poruchových stavů využívá booleovy algebry.

Strom poruchových stavů využívá grafické symboly, které jsou velice podobné symbolům používaných u návrhů elektronických obvodů. Mezi tyto symboly patří především logická hradla OR a AND, které jsou dále popsána v kapitole (1.4). Dalšími hlavními symboly stromu poruchových stavů jsou události. Události reprezentují jednotlivé příčiny a události, které přispívají ke vzniku nežádoucí vrcholové události. Obvykle se stanovuje i pravděpodobnost vzniku ostatních událostí, aby se projevila poruchovost komponent, které nejvíce zvyšují pravděpodobnost vzniku vrcholové události. Mezi události patří vrcholová, normální a základní událost, které jsou rovněž popsány v kapitole (1.4). [10]

1.4 Používané symboly

Metoda FTA využívá pro své výpočty typy událostí a hradel (OR, AND a K/N), které jsou detailněji popsané v evropské normě ČSN EN 61025:2007 (01 0676). [1] V následující části budou popsány základní používané události a to: vrcholová událost (top event), normální událost (normal event) a základní událost (basic event). Dále budou popsána tři základní hradla používaná metodou FTA. Jedná se o hradla OR, AND a K/N. [1]

1.4.1 Vrcholová událost

Jedná se o událost, která je předmětem zájmu, pod kterou se rozvíjí strom poruchových stavů. Tato událost se často označuje jako konečná nebo vrcholová událost. Jde o výstup všech vstupních událostí nebo jejich kombinací. Vrcholová událost je zde reprezentována jako kořen stromu, od kterého se odvíjí další komponenty, mezi které patří normální a základní události a hradla OR, AND a K/N.

1.4.2 Normální událost

Stejně jako u vrcholové události se jedná o událost, která se následně rozvíjí pomocí základních událostí. Oproti základní události neobsahuje žádné vstupní hodnoty. Jde o událost, která je zapříčiněná předchozími událostmi nebo jejich kombinacemi.

1.4.3 Základní událost

Jedná se o událost, která již není dále rozvíjená a nachází se na nejnižší úrovni dané větve stromu. Tato událost obsahuje vstupní hodnoty λ (lambda) a μ (mí). Pomocí kterých se vypočítá pravděpodobnost bezporuchového provozu R(t). Mezi další vstupní hodnoty základní události patří hodnoty MTTF (střední doba do poruchy) a MTTR (střední doba do obnovy). Samotné hodnoty a vztahy mezi nimi jsou popsány v kapitole (1.5).

1.4.4 Hradla

Komponenty, které se používají ke stanovení symbolické logické vazby mezi výstupní událostí a odpovídajícími vstupy. Hradla využívají pro stanovení této vazby booleovu algebru. Metoda FTA využívá k propojení událostí logická hradla, v následujících kapitolách jsou popsána pouze tři základní hradla a to hradla typu OR, AND a K/N.

1.4.5 Hradlo OR

Hradlo OR reprezentuje logický součet. Výstup hradla bude pravdivý, pokud bude pravdivý alespoň jeden z jeho vstupů.

1.4.6 Hradlo AND

Hradlo reprezentující logický součin. Výstup hradla bude pravdivý, jestliže budou pravdivé všechny jeho vstupy.

1.4.7 Hradlo K/N

Hradlo je možnou „kombinací“ předchozích dvou hradel. Výstup hradla bude pravdivý, pokud bude pravdivých K-vstupů z N.

1.5 Vstupní a výstupní hodnoty

Mezi vstupní hodnoty patří intenzita poruch – λ, intenzita oprav – μ, střední doba do poruchy – $MTTF$ a střední doba do opravy – $MTTR$. Výstupními hodnotami jsou pravděpodobnost bezporuchového systému – $R(t)$, pravděpodobnost poruchy – $F(t)$, funkce okamžité pohotovosti – $A(t)$ a funkce okamžité nepohotovosti – $U(t)$. [1] [2] [3]

1.5.1 Vstupní hodnoty

Intenzita poruch

Značíme λ. Je to 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. [2] Vypočítáme ji ze vztahu:

$$\lambda(t) = \frac{1}{MTTF} \ [hod^{-1}] \qquad (1)$$

Intenzita oprav

Značíme μ. Je to 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 na začátku časového intervalu. [2]

$$\mu(t) = \frac{1}{MTTR} \ [hod^{-1}] \qquad (2)$$

Střední doba do opravy

Značíme MTTR (Mean Time To Repair). Očekávaná doba do obnovy. [2]

$$MTTR = \frac{1}{\mu(t)} \tag{3}$$

Střední doba do poruchy

Značíme MTTF (Mean Time To Failure). Očekávaná doba do poruchy. [2]

$$MTTF = \frac{1}{\lambda(t)} \tag{4}$$

1.5.2 Výstupní hodnoty

Pravděpodobnost bezporuchového provozu

Pravděpodobnost, že objekt může plnit požadovanou funkci v daných podmínkách a na daném časovém intervalu (t_1, t_2). [2]

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

Obecně se předpokládá, že objekt je na začátku časového intervalu ve stavu schopném plnit požadovanou funkci.

Pravděpodobnost poruchy

FTA používá pro své výpočty také pravděpodobnostní doplněk k pravděpodobnosti bezporuchového provozu, který se nazývá pravděpodobnost poruchy $F(t)$ [1] a vypočítáme ho pomocí vzorce:

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

Funkce okamžité pohotovosti

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. [2]

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

Funkce okamžité nepohotovosti

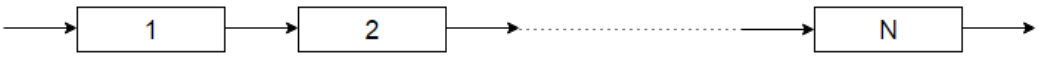
Pravděpodobnost, že objekt není ve stavu schopném plnit v daných podmínkách a v daném časovém okamžiku požadovanou funkci, za předpokladu, že požadované vnější prostředky jsou zajištěny. [2]

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

1.6 Konfigurace systému

1.6.1 Sériová konfigurace systému

V FTA je tento systém reprezentován hradlem OR. Jedná se tedy o systém, který je provozuschopný pouze v případě, jestliže komponenta 1 i komponenta 2 i všechny ostatní komponenty jsou provozuschopné. [1]



Obrázek 1: Ukázka sériového zapojení systému

Pravděpodobnost bezporuchového provozu sériově zapojeného systému vypočítáme pomocí vzorce.

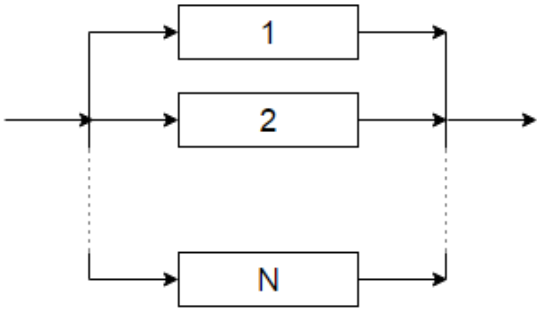
$$R_S(t) = R_1(t) \cdot R_2(t) \cdot \ldots \cdot R_N(t) = \prod_{i=1}^{N} R_i(t) \tag{9}$$

Systém má poruchu v případě, jestliže komponenta 1 nebo komponenta 2 nebo jakákoli jiná komponenta má poruchu. Pravděpodobnost poruchy sériově zapojeného systému vypočítáme pomocí vzorce.

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

1.6.2 Paralelní konfigurace systému

V FTA je tento systém reprezentovaný hradlem AND. Jedná se tedy o systém, který je provozuschopný v případě, jestliže komponenta 1 nebo komponenta 2 nebo jakákoliv jiná komponenta zůstává provozuschopná. [1]



Obrázek 2: Ukázka paralelního zapojení systému

Pravděpodobnost bezporuchového provozu paralelně zapojeného systému vypočítáme pomocí vzorce.

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

Systém má poruchu v případě, že komponenta 1 i komponenta 2 i všechny ostatní komponenty mají poruchu. Pravděpodobnost poruchy paralelně zapojeného systému vypočítáme pomocí vzorce.

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

1.6.3 Zapojení pomocí hradla K/N

Jedná se o systém, který je provozuschopný v případě, jestliže zůstalo alespoň *k* z *n* vstupů provozuschopných. Pravděpodobnost bezporuchového provozu vypočítáme pomocí vzorce. [1]

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

Pravděpodobnost poruchy při zapojení K/N vypočítáme pomocí vzorce.

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

2 Srovnání existujících komerčních softwarů

V této kapitole budou popsány hlavní výhody a nevýhody již existujících programů pro výpočty spolehlivostí systému za pomoci metody FTA. Pro srovnávání byly vybrány následující softwary:

- Fault Tree Analyser

  http://www.fault-tree-analysis-software.com/fault-tree-analysis?type=Aerospace

- RAM Commander

  http://aldservice.com/Download/download-reliability-and-safety-software.html

- ITEM ToolKit

  http://www.itemsoft.com/download_demo.html

- PTC Windchill Quality Solutions

  http://www.ptc.com/product-lifecycle-management/windchill/free-trial

- Isograph

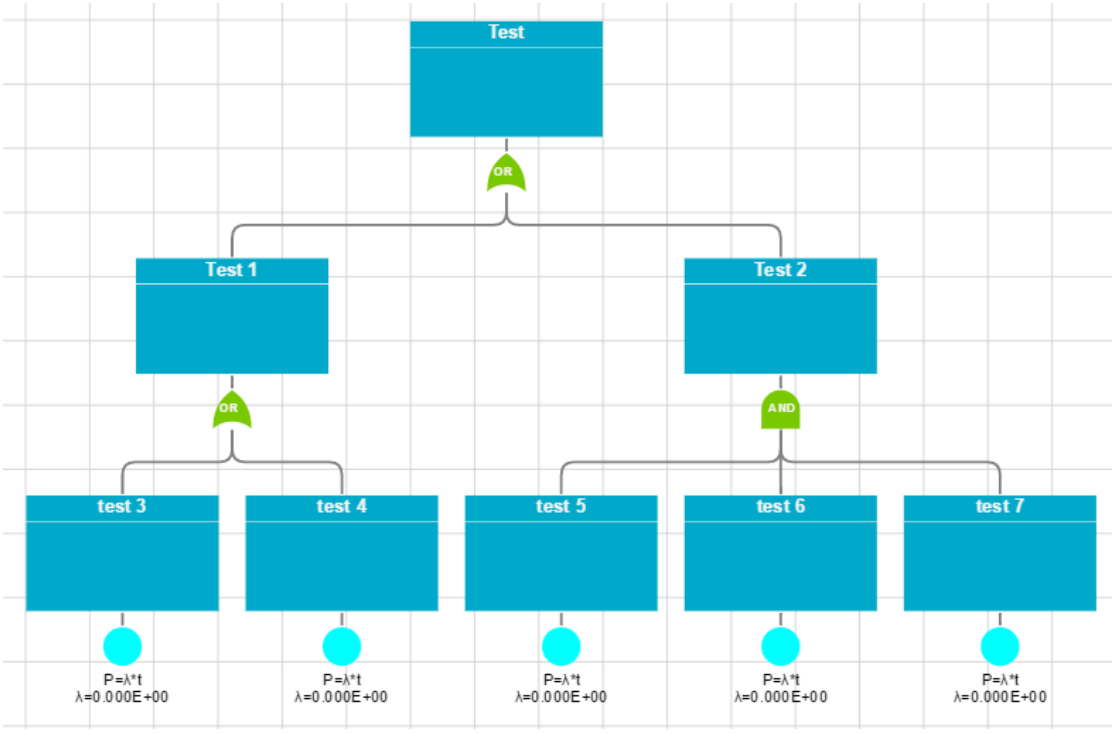
  http://www.isograph.com/software/reliability-workbench/fault-tree-analysis/

2.1 Fault Tree Analyser

Tento software vytvořila firma ALD, jedná se o firmu v oblasti inženýrské spolehlivosti a analýzy a dále bezpečnostní analýzy a řízení. [6] Pro vytvoření nového projektu je potřeba se zaregistrovat. Registrace je bezplatná a stačí vyplnit požadovaný formulář.

2.1.1 Rozhraní softwaru

Po spuštění stránky je zobrazen příklad, ve kterém je již nastavena vrcholová událost, u všech hradel je nastaven typ (u tohoto příkladu jsou využity hradla OR a AND), a základní události mají nastavené hodnoty λ (failure rate).

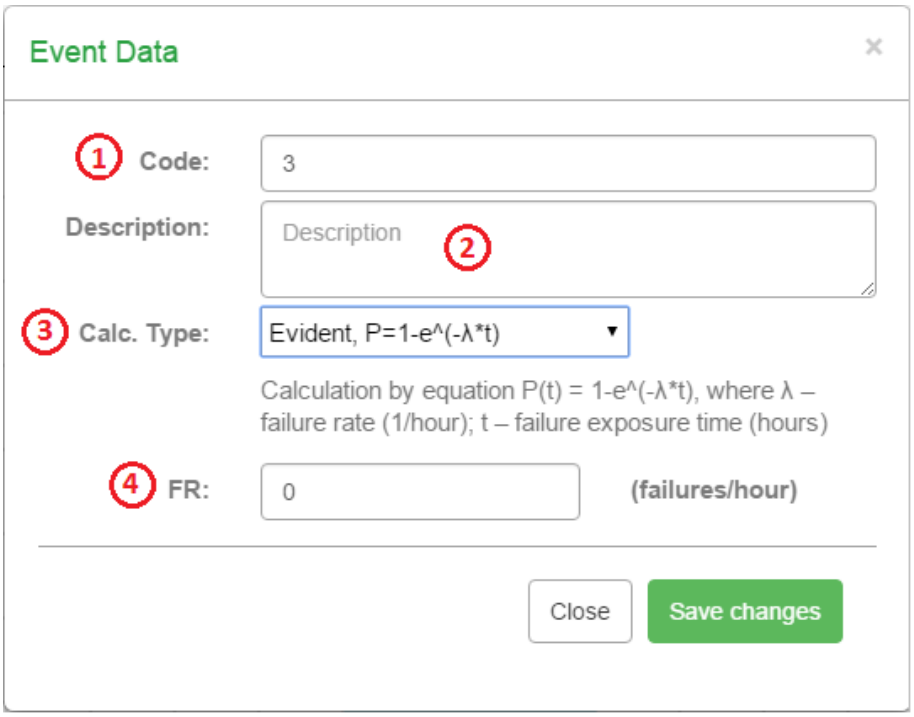


Obrázek 3: Ukázka FTA stromu vytvořeného pomocí aplikace Fault Tree Analyser

Po vytvoření nového projektu nám program nabídne vytvoření hlavní události. V případě, že chceme přidávat nová logická hradla nebo nové události, u kterých máme na výběr z již existujících nebo si můžeme vytvořit novou, musíme si označit hradlo nebo událost, za kterou se nové hradlo nebo událost vytvoří.

V případě, že nechceme danou větev stromu už dále rozvíjet, přidáme novou událost. U které si musíme zvolit, co se u dané události bude počítat, a dále vložíme vstupní hodnoty.

Program umožňuje zobrazení tabulky všech událostí i s jejich vstupními hodnotami a tabulky všech logických hradel.



Obrázek 4: Okno pro nastavení vlastností nové události v programu Fault Tree Analyser

1 – název objektu

2 – popis objektu

3 – typ výpočtu (co bude daná událost počítat)

4 – potřebné hodnoty pro výpočet

2.1.2 Hodnocení programu

Jednoznačnou výhodou tohoto programu oproti zbývajícím je to, že je online. Tudíž není nutné nic stahovat ani instalovat. Nevýhodou je to, že firma je vlastníkem dat. Může nastat problém v řadě zakázek, pokud vlastníkem je firma zhotovující software.

2.2 RAM Commander

Stejně jako Fault Tree Analyser jde o software vytvořený firmou ALD. Předchozí software je určen pouze pro metodu FTA. Zde se jedná o komplexní softwarový nástroj pro spolehlivostní, udržovatelnou analýzu a jejich predikci, optimalizaci náhradních dílů, FMEA/FMECA, testovatelnost, posuzování bezpečnosti a FTA. Jeho

spolehlivostní a bezpečnostní moduly pokrývají všechny všeobecně používané spolehlivostní standardy a analýzy poruch.

2.2.1 Instalace softwaru

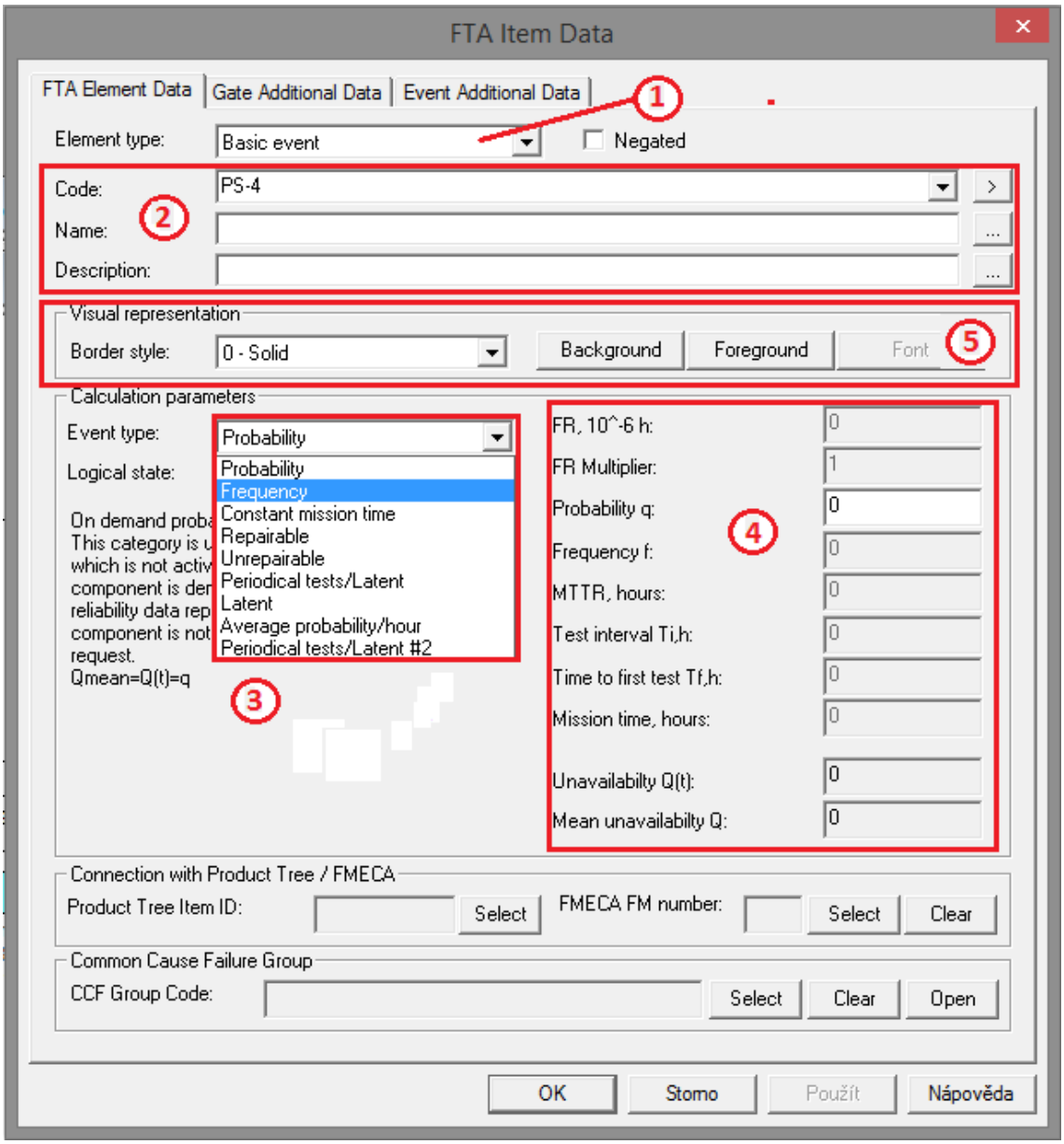
Instalátor tohoto softwaru najdeme na oficiálních stránkách (http://aldservice.com) společnosti ALD. Je zde na výběr ze tří možných verzí:

- FREE DEMO
- STUDENT VERSION
- FULL VERSION

2.2.2 Rozhraní softwaru

Po uvítací obrazovce nám program nabídne mnoho modulů jako například predikce spolehlivosti, spolehlivostní analýzy a analýzy selhání.

Po vybrání modulu FTA máme na výběr ze tří tutoriálů, nebo si můžeme nový diagram vytvořit. Stejně jako u předchozího softwaru od firmy ALD je po vytvoření nového projektu na plátně automaticky vytvořena hlavní událost. Pro přidání další události nebo hradla je potřeba znovu označit událost nebo hradlo, za které se námi vytvořená událost přidá. Při výběru události (Basic) musíme opět zvolit vstupní hodnoty. Program je intuitivní a po výběru typu události se nám aktivují políčka pro vložení daných vstupních hodnot.



Obrázek 5: Okno pro nastavení nové události v programu RAM Commander

1 – nastavení typu objektu

2 – jméno a popis objektu

3 – výstupní hodnoty objektu

4 – vstupní hodnoty objektu

5 – nastavení grafického zobrazení objektu

2.2.3 Hodnocení softwaru

Program je komplexní, obsahuje mnoho modulů (2.2) pro vytvoření spolehlivostní analýzy, i pro analýzy poruch.

2.3 ITEM ToolKit

2.3.1 Instalace softwaru

Instalátor softwaru nalezneme na stránkách společnosti ItemSoft (http://www.itemsoft.com/download_demo.html). Pro stažení je nutné vyplnit dotazník. Pro instalaci je potřeba autorizační klíč, který mi byl zaslán po předchozí komunikaci se zástupcem společnosti.

2.3.2 Rozhraní softwaru

Po spuštění softwaru a vytvoření nového projektu musíme na hlavním panelu přidat požadovaný modul. Na výběr máme z modulů RBD, ETA, FMECA, FTA, MKV. Po výběru modulu FTA se nám zobrazí pracovní plocha s vrcholovou událostí.

Pro přidání nového objektu klikneme pravým tlačítkem na objekt, který bude předkem námi vkládaného objektu. Zvolíme možnost *Add* a vybereme požadovaný objekt. Nový objekt se automaticky zařadí za označený objekt. Pro přidání objektu můžeme zvolit i možnost, že si objekt vybereme z nabídky na hlavním panelu a klikneme na objekt, který bude předkem.

Pro nastavení vstupních hodnot u základních událostí (basic event) si označíme událost a vybereme možnost *Event parametres*. V nově otevřeném okně máme možnost zvolit například typ (základní, normální,…) a název události. Pro vložení vstupních hodnot přejdeme na záložku *Failure*, kde vybereme typ vstupních hodnot. Po zvolení typu vložíme vstupní hodnoty do příslušných textových polí.

Po nastavení všech vstupních hodnot zbývá provést výpočet. Výpočet provedeme pomocí tlačítka na hlavním panelu *Start FT Analysis* nebo v menu nabídce v záložce *Analysis – Perform*. V případě, že výpočet neproběhl v pořádku, nás program upozorní na chyby. Po úspěšném výpočtu přepneme do záložky *Result*, ve kterém máme zobrazené všechny výsledky. Ať už pro vrcholovou událost nebo pro všechny použitá hradla.

2.3.3 Hodnocení softwaru

Výhodou softwaru je jeho přehlednost v návrhovém prostředí. A jeho přidávání nových objektů. Nevýhodou programu je naopak nastavení základních událostí. Při kterém se musí uživatel „proklikat“ záložkami, než se dostane přímo k nastavení

hodnot. Při nastavení vstupních hodnot musí uživatel vyplnit také název poruchového modelu.

2.4 PTC Windchill Quality Solutions

2.4.1 Instalace softwaru

Instalátor softwaru nalezneme na stránkách společnosti PTC (http://www.ptc.com/product-lifecycle-management/windchill/free-trial). Jedná se o 30denní trial verzi. Pro stažení je nutné vyplnit formulář. Ten vyžaduje jak osobní informace, tak i informace o tom, na co se bude software využívat.

2.4.2 Rozhraní softwaru

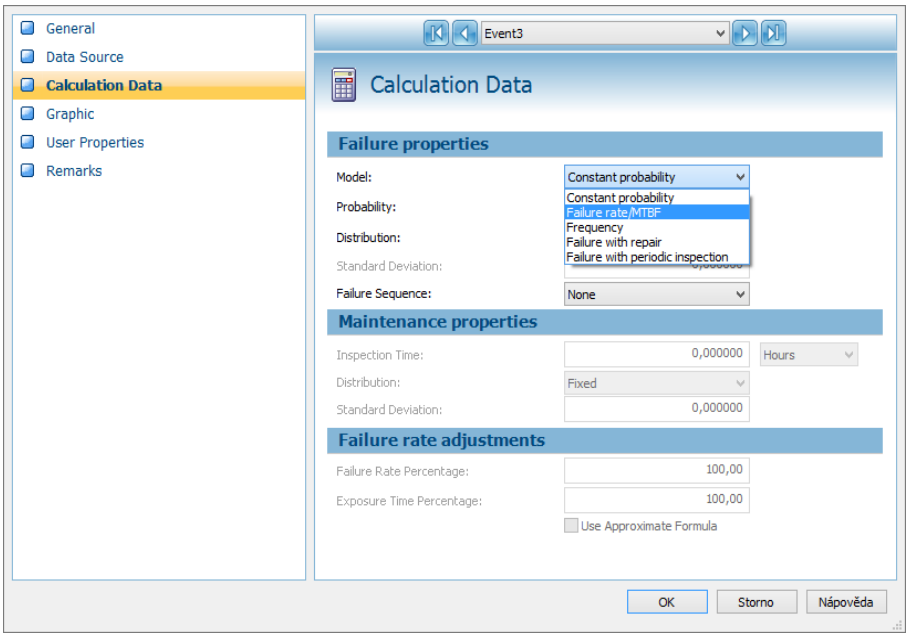
Po spuštění softwaru se nám zobrazí úvodní obrazovka, na které si můžeme vybrat z modulů RBD, Markov, FMEA, LCCA, FTA, apod. Po vybrání modulu FTA si vytvoříme nový projekt.

Nyní máme na pracovní ploše vrcholovou událost. Pro přidání nového hradla nebo události máme na výběr ze dvou možností. Klikneme pravým tlačítkem na předka a zvolíme možnost *Insert Input* a v novém okně si vybereme požadovaný objekt. Nebo, na hlavním panelu vybereme možnost *Insert*, kde máme dále na výběr, možnosti *Gate* a *Event* a opět vybereme z nabízených objektů.

Pro nastavení vstupních hodnot u základních událostí (basic event) si označíme událost a vybereme možnost *Properties*. V nově otevřeném okně zvolíme z postranní nabídky možnost *Calculation Data*.

Po nastavení všech vstupních hodnot zvolíme možnost *Calculate*, která se nachází na hlavní liště nebo v menu *System – Calculate*. Opět se nám otevře okno, ve kterém zvolíme, pro jaký model chceme vytvořený strom vypočítat. V postranním menu v záložce *FTA – General* můžeme nastavit, co chceme pro vrcholovou událost vypočítat. Spolehlivost, frekvenci, počet selhání apod. Dále potom pro jaké časy tyto hodnoty chceme počítat a jestli chceme tyto hodnoty vypočítat pouze pro vrcholovou událost nebo pro všechny hradla.

Po nastavení všech kritérií a následném výpočtu se nám zobrazí tabulka výstupních hodnot, které jsme si zadali. Tuto tabulku si můžeme vytisknout nebo převést do Excelu.



Obrázek 6: Okno pro nastavení vstupních hodnot v programu PTC

2.4.3 Hodnocení softwaru

Výhodou tohoto softwaru je jeho jednoduchost. Intuitivní vkládání nových objektů i vstupních hodnot. Tento software má nejpřehlednější uživatelské rozhraní ze všech testovaných softwarů. Nevýhodou by pro některé uživatele mohlo být, že nemůžou s objekty hýbat. Dle mého názoru se jedná o dobrou vlastnost. Jelikož program sestavuje FTA strom do přehledné formy a zásah do této formy by mohl strom znepřehlednit.

2.5 Isograph

Software od společnosti Isograph pro výpočet spolehlivosti obsahuje např. moduly FMECA, FMEA, FTA, RBD a modul pro Markovovu analýzu. Bohužel tento software jsem nemohl otestovat, jelikož jsem neobdržel žádný licenční klíč pro tento software.

2.6 Porovnání softwarů

	Fault tree analyser	RAM Commander	Item Toolkit	PTC	Vlastní program
Používaná hradla	AND, OR, K/N	AND, OR, K/N, XOR, NAND, NOR	AND, OR, K/N	AND, OR, K/N, XOR, NAND, NOR	AND, OR, K/N
Grafický výstup	Ne	Ano	Ano	Ano	Ano
Textový výstup	Ano	Ano	Ano	Ano	Ano
Omezení hradel	50 online	15 (demo verze)			
Periodická údržba	Ano	Ne	Ano	Ano	Ano

3 Program

V kapitole 3 bude vysvětlen princip běhu programu, který řeší spolehlivost systémů pomocí metody FTA. První část této kapitoly je věnována tvorbě samotného stromu poruchových stavů, který si uživatel sám tvoří pomocí grafického rozhraní programu. V další části této kapitoly je popsán samotný algoritmus pro výpočet pravděpodobnosti bezporuchového provozu $R(t)$.

3.1 C#

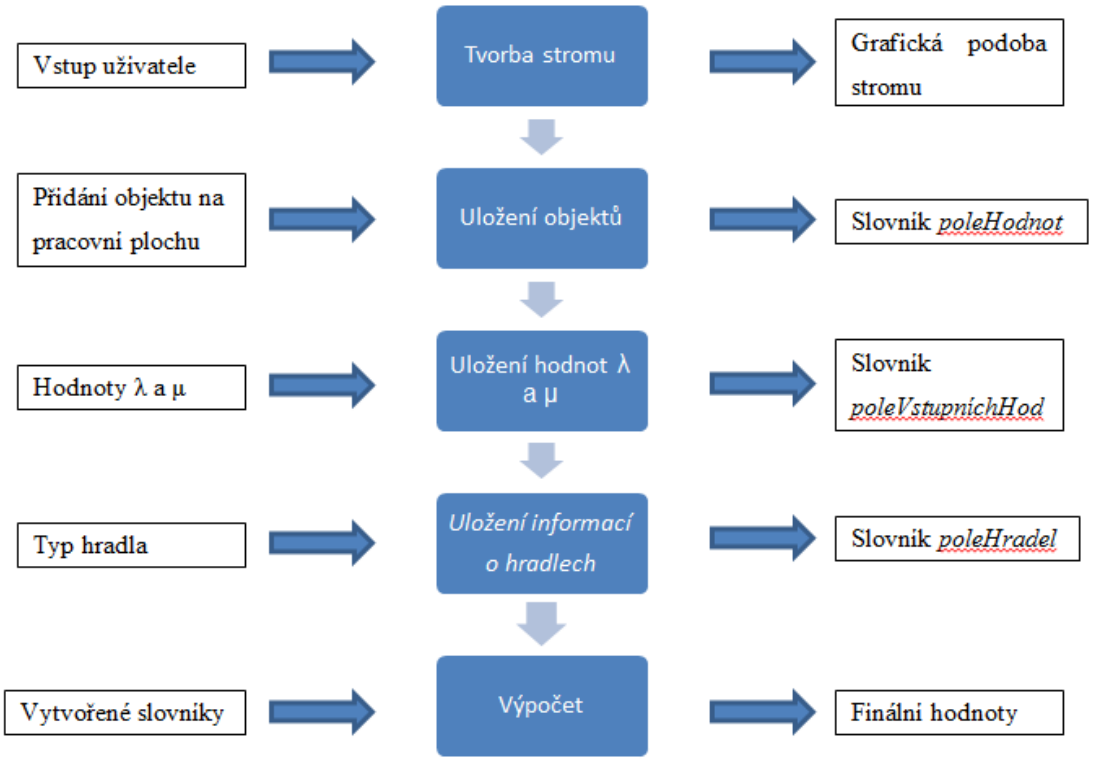
Program je napsán pomocí programovacího jazyka C#. Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk. Kromě desktopových aplikací ho lze využít také na tvorbu webových aplikací, databázových programů nebo pro mobilní aplikace. Programovací jazyk jsem vybral z důvodu přehledného vývojového prostředí a práce s formuláři a objekty.

Program se skládá z hlavní třídy *Main*, která řídí všechny ostatní třídy. Pro vytváření stromu poruch třída využívá třídu *Udalost*, která obsahuje základní informace o vytvořených objektech. Např. zda se jedná o hradlo či událost, jejího předka,… Třída obsahuje také metodu pro náhodné generování stromu poruch.

Od třídy *Udalost* dědí vlastnosi a parametry další dvě třídy, a to *Gate* a *Evnt*. Třída *Evnt* navíc obsahuje informace o typu události (Basic, Normal), v případě, že se jedná o základní událost, také informace o vstupních parametrech. Třída *Gate* obsahuje informace o typu hradla (AND, OR, K/N) a v případě, že se jedná o hradlo K/N, také hodnoty *k* a *n*.

Poslední třídou je *Vypocet*. Tato třída se stará o jednotlivé výpočty, které hlavní třída využívá pro získání hodnot $R(t)$ – pravděpodobnost bezporuchového systému, $F(t)$ – pravděpodobnost poruchy, $A(t)$ – funkce okamžité pohotovosti a $U(t)$ – funkce okamžité nepohotovosti. [13] [14]

3.2 Princip chodu programu



Obrázek 7: Blokový diagram chodu programu, popis vstupů a výstupů jednotlivých bloků

3.3 Tvorba stromu

Při každém spuštění aplikace se automaticky vytvoří hlavní událost, kde základním výstupem programu budou hodnoty:

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

Uživatel vybírá jednotlivé objekty, které chce vložit na pracovní plochu. Může přidávat hradla a události, které jsou v nabídce. Na výběr má ze dvou událostí, kterými jsou základní událost (BASIC) a normální událost. Kde základní událost je konečná událost a už nemůže být dále rozvíjena, oproti tomu normální událost je dále rozvíjena. Dalším rozdílem mezi základní a normální událostí je to, že u základní události uživatel

nastavuje hodnoty λ a μ, nebo MTTR a MTTF. U normální události se žádné hodnoty nenastavují.

Dalším objektem, který je na výběr, je hradlo. Typ hradla si nastaví uživatel až při editaci hradla. Na výběr má z typů hradel OR, AND nebo K/N.

Další možností bylo přidat všechny tyto hradla do seznamu dostupných objektů pro vkládání na pracovní plochu. První zmíněnou možnost byla zvolena z toho důvodu, kdyby se uživatel spletl a zvolil pro danou větev stromu špatné hradlo. V tom případě by hradlo musel smazat, a tím by se smazali všechny komponenty, hradla a události od daného hradla dál. Nyní ale uživatel pouze změní typ hradla bez jakýchkoliv jiných zásahů do stromu.

Posledním objektem, který může uživatel přidat na pracovní plochu je zkopírovaný objekt. Tento objekt může obsahovat všechny dostupné objekty, kromě vrcholové události. Do tohoto objektu si při vytváření stromu poruch může uživatel zkopírovat právě vybraný objekt. Při následném vložení zkopírovaného objektu se na pracovní plochu vloží daný objekt se stejnými nastavenými hodnotami, které obsahoval zkopírovaný objekt.

V případě, že chce uživatel vložit na pracovní plochu hradlo nebo události, musí si nejdříve zvolit předka vkládaného objektu. Toto zvolí kliknutím na objekt, který má být předkem. Jestliže chce uživatel vložit hradlo, musí mít označenou událost a naopak. Program uživatelovi nedovolí, aby za událost mohl vložit jinou událost. V případě, že uživatel vybere jako předka událost, se tlačítka pro vkládání událostí stanou neaktivními. To samé platí i pro hradla. Tato funkčnost byla zvolena z důvodu, aby mohl uživatel vytvořit požadovaný strom. Při označení předka vkládané události se automaticky aktivují objekty, které je možné použít. Po vložení objektu na pracovní plochu se nově vložený objekt automaticky propojí se svým předkem.

U hradla má uživatel k dispozici na výběr ze tří možností. Může si vybrat hradlo typu OR, AND nebo K/N. U posledního zmiňovaného hradla ještě musí uživatel nastavit hodnoty k a n. Kde k představuje počet objektů, které jsou povolené a n reprezentuje počet všech objektů, které vstupují do tohoto hradla. Hodnota k musí být menší nebo rovna hodnotě n. V případě, že $k = 1$, bude způsob zálohování stejný, jako kdyby si uživatel vybral hradlo typu AND (musí být v provozuschopném stavu alespoň jedna komponenta z n). Jestliže $k = n$, tak se bude jednat o stejný případ, jako kdyby

uživatel vybral hradlo typu OR (v provozuschopném stavu musí být všechny komponenty). Dalším objektem, který může uživatel ovlivňovat změnou vstupních parametrů je základní událost.

U základní události musí uživatel nejprve zvolit, jaké hodnoty bude chtít zadat. Na výběr má z dvojice hodnot λ, μ a *MTTR*, *MTTF*. Tyto hodnoty nemohou být menší než nula. Pro tento případ je program také ošetřen a v případě, že bude hodnota menší než 0, je uživatel upozorněn.

U všech objektů, kromě hradel, může uživatel nastavit název a popis dané události. Název slouží pro jednoduchou orientaci, na co daná událost slouží. Pole popis slouží pro rozsáhlejší popis dané události.

3.4 Ošetření chyb v programu

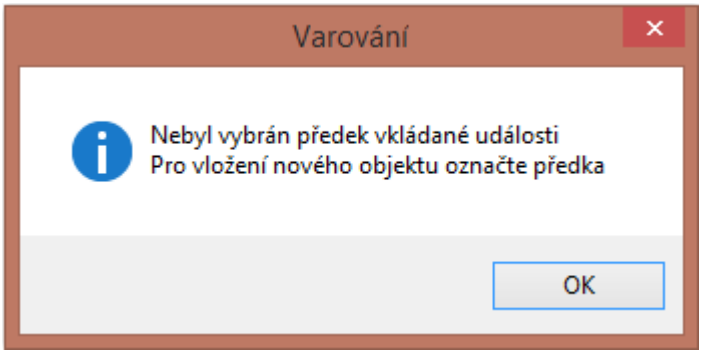
V programu se nachází několik míst, kde může uživatel vytvořit chybu. Tyto jednotlivé chyby jsou v programu ošetřeny hned na daných místech, tzn. u textových polí, kde uživatel zadává vstupní parametry. Vstupy jsou ošetřeny pomocí:

- již dostupných informací, např. u hradel *K/N* je hodnota *n*, která reprezentuje počet všech přímých potomků tohoto hradla, se nastavuje automaticky v závislosti na již zmíněném počtu přímých potomků
- regulárních výrazů (ošetření vstupních parametrů u základních událostí)

V případě, že uživatel zadá špatné vstupní hodnoty, program ho ihned upozorní na chybu. Jestliže uživatel chybu neopraví, nebude moci dále pokračovat v práci a program jeho veškeré nastavené parametry neuloží. [11]

Chyba	Popis chyby	Upozornění
Neoznačení předka	Při vytváření stromu poruch, není vybrán předek vkládané události nebo hradla	Program uživatele upozorní, že není vybrán předek a poradí, jak má uživatel postupovat
Parametr K > N	Při zadávání parametrů u hradla K/N, uživatel zadá hodnotu K vyšší než N	Uživatel je programem upozorněn na chybu a zároveň je nabídnuta rada jak dále postupovat
Záporné vstupní parametry u základních událostí	Při zadávání vstupních parametrů u základních událostí, kdy uživatel zadá záporné číslo	Program uživatele upozorní, že vstupní parametry nesmějí být záporné
Pokus o kopírování vrcholové události	Uživatel se pokouší zkopírovat do schránky vrcholovou událost. To není možné, jelikož vrcholová událost je pouze jedna	Uživatel je upozorněn, že vrcholovou událost není možné zkopírovat
Alfanumerické znaky u vstupních parametrů	Uživatel zadává do polí pro vstupní parametry alfanumerické znaky	Program uživatele upozorní, že vstupní parametry musí být pouze čísla

Tabulka 1: Přehled chyb, vznikajících při tvorbě stromu poruch



Obrázek 8: Okno s chybou vznikající při tvorbě stromu poruch

3.5 Uložení objektů

Pro ukládání jednotlivých informací o objektech byla použita datová struktura *Dictionary*, protože umožňuje ukládat hodnotu pod daný klíč. V tomto případě je klíčem *id* dané události.

Při každém umístění nové události nebo hradla na pracovní plochu jsou uloženy informace o daném objektu do slovníku *udalosti*. Struktura slovníku vypadá následovně:

$$udalosti = \{klíč: id\ objektu, hodnota: \{Objekt\ Udalost\}\}$$

Kde klíč reprezentuje *id* nového vloženého objektu a hodnota obsahuje objekt *Udalost*, jejíž struktura vypadá následovně:

$$Udalost = (id, predek, pozice\ X, pozice\ Y, jméno, sirka, vyska, název, popis)$$

- *id* reprezentuje nový objekt,
- *predek* značí objekt, který ve stromové struktuře je přímý předek daného objektu,
- pozice *X* a *Y* jsou souřadnice objektu na pracovní ploše,
- jméno objektu značí, zda je objekt hradlo nebo událost,
- *sirka* a *vyska* reprezentují velikost objektu na pracovní ploše,
- *název* a *popis* reprezentují textový popis daných událostí. V případě, že se jedná o základní událost, nachází se v konstruktoru třídy *Udalost* i parametr *polomer*, který se využívá při vykreslení objektu základní události na pracovní plochu.

Pro ukládání vypočtených hodnot u všech objektů slouží slovník *poleHodnot*. Do tohoto slovníku se ukládá každý nově vložený objekt a ukládají se do něj všechny počítané finální hodnoty, které si uživatel při výpočtu zvolí. Na výběr má z *R(t), F(t), A(t)* a *U(t)* (viz kapitola 1.5.2). Struktura slovníku vypadá následovně:

$$poleHodnot = \{klíč: id\ objektu, hodnota: \{klíč: t, hodnota: R(t)\}\}$$

Klíč slovníku je reprezentován jako *id* vkládaného objektu a hodnota obsahuje další slovník. Klíč tohoto vnitřního slovníku reprezentuje čas *t* a hodnota je vypočítaná finální hodnota, která byla zvolena při výpočtu.

Další možností byla volba datové struktury *List*. Tato struktura ukládá hodnoty podle indexu od 0. Pro program by se toho moc nezměnilo a všechno by fungovalo stejně jako v případě použití datové struktury *Dictionary*. Problém by nastal, kdyby se uživatel rozhodl smazat část stromu. V tom případě by se musely smazat i všechny hodnoty z *Listu* na daných indexech. A právě zde by nastal problém.

Po vymazání objektů by se všechny zbývající objekty v seznamu posunuly o právě smazaný počet. Složitě by se musely upravovat *id* zbývajících objektů i jejich propojení. Proto jsem zvolil datovou strukturu *Dictionary*, u které se tento problém nemusí řešit. Zde, v případě úpravy stromu a případného mazání hodnot ze slovníku, se nemusí nic upravovat. Pouze se vymaže položka ze slovníků, kde klíč je roven *id* daného objektu.

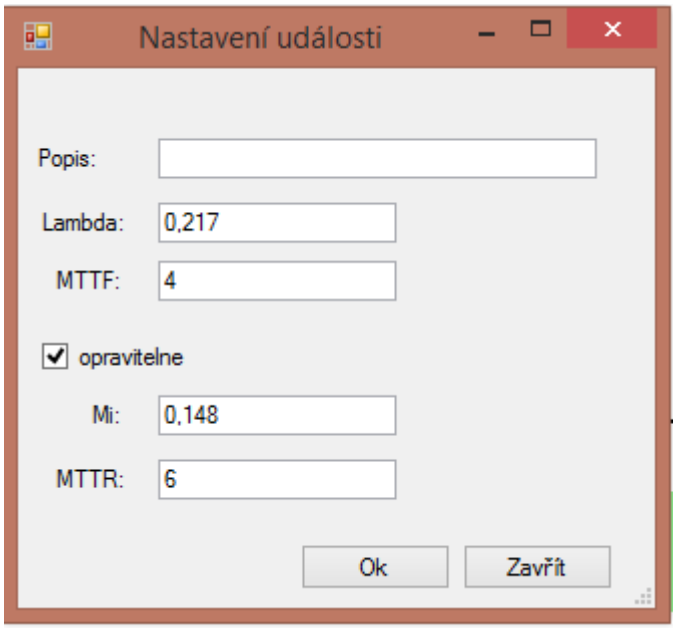
3.6 Uložení hodnot λ a μ

Jak již bylo zmíněno v kapitole (1.4.3), základní událost je jednou ze dvou objektů, které může/musí uživatel ovlivnit. V tomto případě je to zadávání vstupních hodnot λ, μ nebo *MTTF, MTTR*, které se dále používají pro další výpočty. V případě zvolení dvojice vstupních hodnot *MTTF* a *MTTR* se tyto hodnoty převedou na hodnoty λ a μ pomocí vzorců (1)(2). Jak už z předchozí kapitoly víme, každý nově vytvořený objekt se uloží do slovníku pod svým *id*. To platí i pro základní událost. Ta navíc kvůli zmiňovaným hodnotám λ a μ využívá další datovou strukturu *Dictionary* pro ukládání těchto hodnot. Struktura slovníku vypadá následovně:

$$poleVstupnichHodnot = \{klíč: id\ objektu, hodnota: Tuple < \lambda, \mu, isLambdaMi >\}$$

Klíč slovníku je reprezentován jako *id* dané základní události a hodnota slovníku obsahuje datovou strukturu *Tuple*, která obsahuje 3 hodnoty. První reprezentuje hodnotu

λ, druhá μ a třetí hodnota je pomocná pro zpětné nastavení dané události. V případě, že uživatel zadal vstupní hodnoty *MTTR* a *MTTF*, tak se mu při následné editaci stejného hradla hodnoty λ a μ přepočítají zpět na hodnoty *MTTR* a *MTTF*.



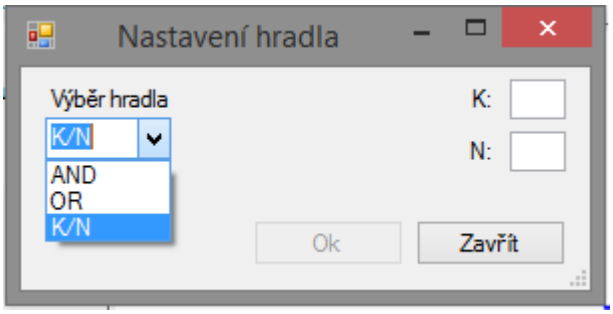
Obrázek 9: Okno pro nastavení vlastností základní události programu

3.7 Uložení informací o hradlech

Stejně jako hodnoty u základních událostí, tak i vlastnosti jednotlivých hradel musí uživatel nastavit. U hradel se nastavuje typ hradla, kde je na výběr z typů OR, AND a K/N. V případě výběru hradla K/N musí uživatel vyplnit hodnoty *k* a *n* jak bylo zmíněno v kapitole (3.3). Stejně jako u základní události, i zde používám pro uložení informací o hradlech datovou strukturu *Dictionary*. Její struktura vypadá následovně:

$$poleHradel = \{klíč: id\ objektu, hodnota: Tuple < typ, k, n >\}$$

Klíč slovníku je reprezentován jako *id* daného hradla a hodnota slovníku obsahuje datovou strukturu *Tuple*. Do níž se ukládají 3 hodnoty. Jde o typ hradla a dále potom hodnoty *k* a *n*. V případě, že si uživatel vybere jeden z typů hradel OR nebo AND, tyto hodnoty jsou automaticky nastaveny na hodnotu 0.



Obrázek 10: Okno pro nastavení vlastností hradel programu

Po vyplnění všech hodnot u základních událostí a definování všech hradel lze přistoupit k samotnému výpočtu finálních hodnot *R(t)*, *F(t)*, *A(t)* nebo *U(t)* popsaných v kapitole (1.5.2).

3.8 Pomocné datové struktury

Následující kapitola popisuje některé pomocné datové struktury, které při výpočtu finálních hodnot využívám. Většinou se jedná opět o datové struktury typu *Dictionary*.

Jednou z prvních pomocných datových struktur je struktura typu *Dictionary*, jenž nese název *poleVypocteno*. Jedná se o slovník, ve kterém je uložena informace o tom jestli je už daná událost vypočítaná tzn. slovník $poleHodnot$ s příslušných klíčem *id* dané události, obsahuje hodnoty *R(t)*. Struktura slovníku vypadá následovně:

$$poleVypocteno = \{klíč: id\ objektu, hodnota: TRUE/FALSE\}$$

Stejně jako ve všech používaných slovnících, i zde je klíčem opět *id* dané události. Hodnota slovníku nabývá dvou hodnot *true* a *false*. Kde hodnota *true* reprezentuje situaci, kdy hodnoty *R(t)* ve slovníku $poleHodnot$ jsou vypočítány; a hodnota *false* reprezentuje opačnou situaci.

Dalším datová struktura typu *Dictionary* nese název *potomci*. Tento slovník slouží k ukládání informací o potomcích daného objektu. Hodnoty se do slovníku ukládají pokaždé, když uživatel vytvoří novou událost nebo hradlo. Struktura slovníku vypadá následovně:

$$potomci = \{klíč: id\ objektu, hodnota: List(id\ potomků)\}$$

Zde jako klíč ve slovníku slouží *id* objektu. Toto *id* reprezentuje objekt, který je předkem nově vkládaného objektu. Hodnota slovníku obsahuje datovou strukturu *List*, která obsahuje *id* nově vkládaných objektů.

3.9 Generování stromu

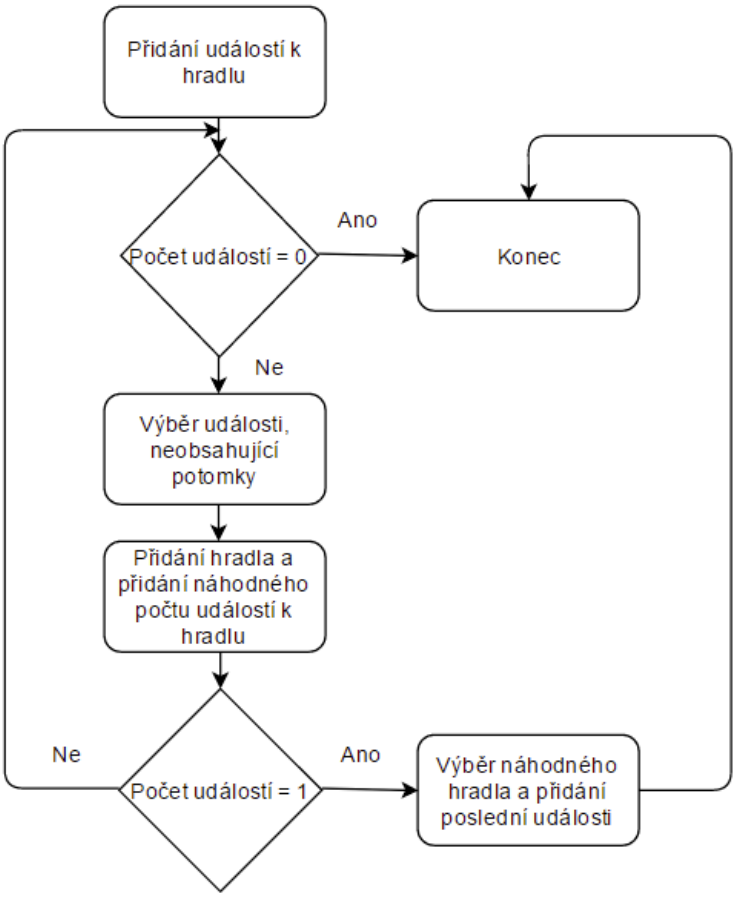
Další možností jak vytvořit strom poruch je automatické generování stromu. Uživatel má možnost si vygenerovat strom poruch. Před generováním musí uživatel zvolit počet hradel a událostí, které budou tvořit daný náhodný strom poruch. Počet událostí musí být minimálně 2× vyšší než je počet hradel (15). Požadavek je z důvodu, aby každé hradlo mělo alespoň dva potomky a tvořilo minimálně binární strom.

$$počet\ událostí \geq počet\ hradel \cdot 2 \qquad (15)$$

3.9.1 Binární strom

Jedná se o datovou strukturu, která se v informatice využívá pro ukládání a vyhledávání dat. Jde o orientovaný graf s jedním hlavním vrcholem (kořen) a několika dalšími vrcholy. Z kořene vede právě jedna cesta do kteréhokoliv vrcholu. Každý vrchol má právě dva potomky.

3.9.2 Algoritmus pro generování stromu



Obrázek 11: Algoritmus pro generování stromu poruch

Krok 1. Nejprve se vytvoří hlavní událost. Jako její potomek se vytvoří hradlo. O jaké hradlo se bude jednat, jestli AND, OR nebo K/N rozhodne náhodný výběr.

Krok 2. Dalším krokem je náhodný výběr hradla, které nemá žádné potomky. Po výběru se tomuto hradlu přidá náhodný počet potomků. Tento počet je zdola omezen minimálním počtem událostí na hradlo (2) a shora omezen počtem událostí – počet hradel × 2 (16).

$$2 \leq počet\ vkládaných\ událostí \leq počet\ hradel \cdot 2 \qquad (16)$$

Po přidání daného počtu událostí je dalším krokem přidání hradla. Proběhne náhodný výběr události, která neobsahuje žádné potomky. K této události je přidáno hradlo a stejně jako u prvního přidaného

hradla, i zde se provede náhodný výběr, o jaké hradlo se bude jednat (AND, OR nebo K/N).

Krok 3. Krok 2 probíhá do té doby, dokud není počet hradel roven 0. Krok 3 probíhá do doby, dokud není počet událostí roven 0 nebo 1. Když se počet událostí rovná 0 algoritmus končí, v případě že je počet událostí roven 1 probíhá poslední krok algoritmu. V tom případě se náhodné hradlo, kterému se poslední událost přiřadí a tím je algoritmus generování ukončen.

Další funkcí generování je možnost náhodného generování vstupních hodnot pro základní události. Před generováním stromu poruch si uživatel může vybrat, jaké vstupní parametry budou základní události obsahovat. Vstupní parametry rozlišujeme podle toho, o jaký systém se bude jednat. Jestli půjde o obnovitelný nebo neobnovitelný systém.

V případě neobnovitelného systému bude mít uživatel na výběr:

- λ - lambda
- MTBF
- Pravděpodobnost poruchy

U obnovitelného systému si uživatel může vybrat mezi parametry:

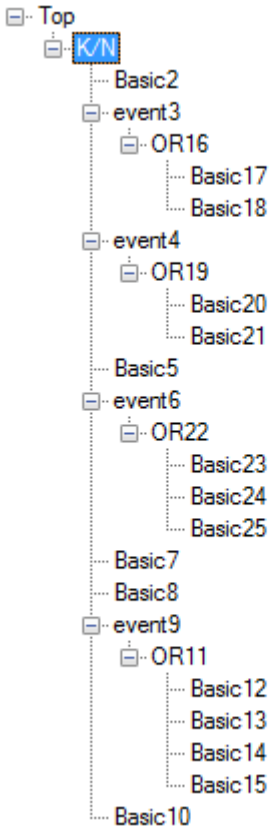
- λ, μ – lambda, mí
- MTBF – MTTR
- Pravděpodobnost poruchy
- Pravděpodobnost obnovy
- Periodu obnovy

V případě zvolení obnovitelného systému si uživatel volí i periodu obnovy. Tato hodnota ukazuje, kdy se daný objekt v průběhu systému obnoví na svou původní hodnotu. Při generování se pro jednotlivé objekty (základní události) vygeneruje náhodná hodnota periody obnovy od 1 do uživatelem zvolené periody obnovy.

Jestliže se bude jednat o hodnoty MTBF nebo MTTR bude si moci uživatel zvolit meze těchto hodnot. V ostatních případech se budou hodnoty pohybovat mezi hodnotami 0 a 1.

I zde je možné po vygenerování stromu poruch ručně přidávat, odebírat nebo editovat jednotlivé objekty.

Uživatel vidí strom poruch jako stromovou strukturu, které je v programu zobrazena pomocí objektu *TreeView* [11] [12] . Uživatel má možnost v tomto objektu jednotlivé události přidávat, upravovat, kopírovat nebo odebírat. V případě přidání nového objektu (hradlo nebo událost), stačí označit objekt, který bude sloužit jako rodič a přidat danou událost nebo hradlo. Při odebírání objektu ze stromu poruch, označí uživatel danou událost nebo hradlo, které chce odstranit a zvolí možnost smazání objektu.



Obrázek 12: Příklad stromu poruch při náhodném generování

Při editaci daného objektu se uživateli zobrazí tabulka (*DataGridView*) [11] [12] , s parametry daného objektu. V případě, že se jedná o hradlo, vidí uživatel pouze popis hradla a zároveň o jaké hradlo se jedná. Jestliže je toto hradlo typu K/N zobrazí se také hodnoty těchto dvou parametrů. Hodnotu K může uživatel měnit, ale pouze v případě, že hodnota K bude menší nebo rovna hodnotě N.

	Name	Value
▶	Popis	
	K	2
	N	9
	Vysledek	0

Obrázek 13: Tabulka parametrů hradla K/N

Při editaci událostí záleží, jestli se jedná o základní nebo o normální událost. V případě normální události je zobrazen pouze popis události. Jestliže se jedná o základní událost, může uživatel nastavit také hodnoty vstupních parametrů (λ, μ, MTTR, MTBF,…).

	Name	Value	Unit
	Popis		-
	Lambda	0,005	Roky^-1
	Mi	0,02	Roky^-1
▶	MTTR	50	Roky
	MTTF	200	Roky
	Perioda údržby	4	Roky
	Vysledek	0	Roky
*			

Obrázek 14: Tabulka vstupních parametrů pro základní události

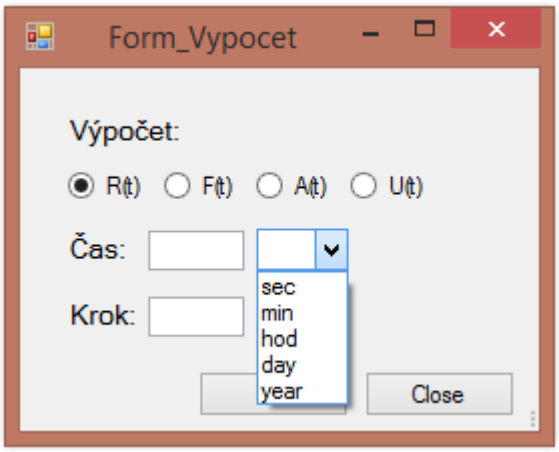
Při kopírování si uživatel zvolí, jakou část stromu poruch chce kopírovat. Po označení daného objektu ve stromě zvolí kopírování. Po zkopírování objektu si následně vybere, do jaké části chce daný objekt vložit. Označí si objekt, který bude předkem zkopírovaného podstromu a vybraný podstrom vloží za označený objekt. Při kopírování se zkopírují také všechny vlastnosti původních objektů.

Další funkcí programu je možnost vykreslení dané části stromu poruch. Při kliknutí na daný objekt se na pracovní plochu tento objekt vykreslí. S ním i všechny jeho potomci až do druhé úrovně od zvoleného objektu. Nejdříve program zjistí, do jaké úrovně je daný podstrom rozšířen. V případě, že počet úrovní je větší než 2, vypočítá všechny potomky na druhé úrovni od zvoleného objektu. Následně se spočítají také všechny objekty na první úrovni od zvoleného objektu. Po vypočítání objektů na daných úrovní se nejdříve na pracovní plochu vykreslí všechny objekty z druhé úrovně, poté objekty z první úrovně a nakonec vybraný objekt.

3.10 Výpočet

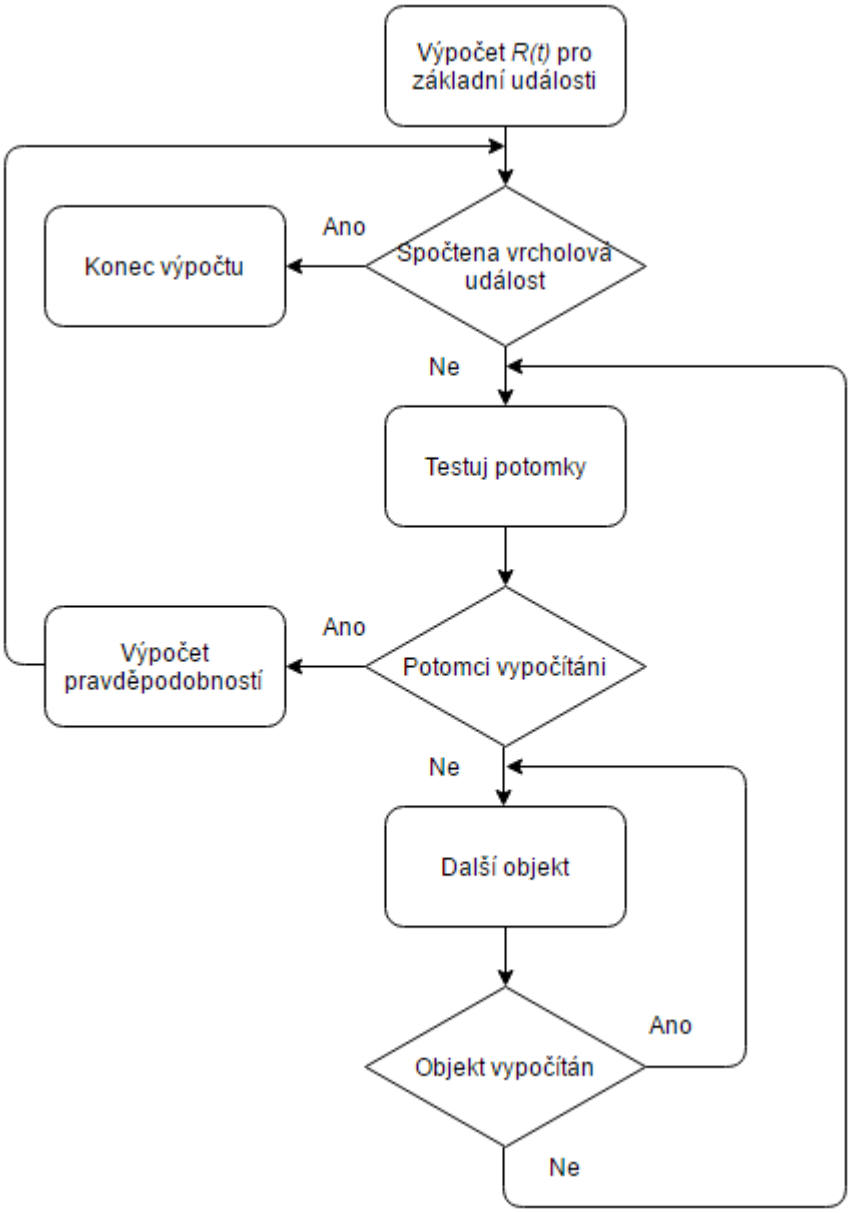
Před samotným výpočtem, musí uživatel nejprve zadat potřebné parametry, podle kterých bude program počítat. Uživatel nastavuje 4 hodnoty parametrů (Obrázek 15). Prvním parametrem, který se nastavuje, je hodnota, kterou bude program počítat. Na výběr jsou možnosti *R(t), F(t), A(t)* a *U(t)*. Všechny tyto hodnoty jsou blíže popsány

v kapitole (1.5.2). Dalším parametrem, který uživatel zadává je simulační čas. S tím souvisí i třetí parametr, kde se zadává, v jakých jednotkách bude zadaný čas. Jsou zde možnosti sekund, minut, hodin, dní, a roků. Poslední parametr, který uživatel nastavuje je, po jakých krocích bude program dané hodnoty počítat.



Obrázek 15: Okno pro nastavení parametrů hlavního výpočtu

Dříve než začne program počítat zadané finální hodnoty, proběhne několik výpočtů, podle hodnot zadaných u základních událostí. Výpočet probíhá v několika krocích, které jsou vyobrazeny na obrázku (Obrázek 16) a tyto jednotlivé kroky jsou popsány v následujících kapitolách.



Obrázek 16: Algoritmus pro výpočet hodnot pravděpodobnosti bezporuchového provozu (prohledávání stromu do hloubky)

3.10.1 Začátek výpočtu

Ještě před začátkem celého algoritmu pro výpočet pravděpodobnosti bezporuchového provozu pro vrcholovou událost, jsou vypočítány tyto hodnoty pro všechny základní události. Pro výpočet se využívá vzorce (5) a definovaných hodnot λ a μ. Po výpočtu hodnot *R(t)* pro danou základní události se v pomocné slovníku *poleVypocteno*, popsaného v kapitole (3.8), změní hodnota na pozici *id* této události z *false* na *true*. Tento postup se provede pro každou definovanou základní událost. Poté následuje samotný algoritmus.

3.10.2 Spočítána základní událost

Jako první program testuje, zda již není pravděpodobnost bezporuchového provozu pro vrcholovou událost spočtena. V případě, že ano, je výpočet ukončen a data jsou uložena ve slovníku $poleHodnot$ a je možno je zobrazit v tabulce pomocí reportů. V opačném případě, tedy pokud není pravděpodobnost spočtena, přistoupí program k dalšímu kroku.

3.10.3 Testování potomků vybraného objektu

Dalším krokem algoritmu je testování, zda jsou potomci daného objektu vypočítáni. Pro tento krok jsou využívány pomocné slovníky $poleVypocteno$ a $potomci$. Nejdříve program vybere, ze slovníku *potomci,* všechny potomky daného objektu, které testuje pomocí slovníku $poleVypocteno$. V případě, že jsou pro tyto objekty vypočítány pravděpodobnosti bezporuchového provozu *R(t)*, může program přestoupit k další části algoritmu. V opačném případě program přistoupí k dalšímu objektu, pro který je hodnota ve slovníku $poleVypocteno$ rovna $false$.

3.10.4 Výpočet hodnot *R(t)*

Jestliže všechny předchozí kroky byly splněny, tudíž pravděpodobnost bezporuchového provozu pro daný objekt nebyla spočtena, ale byla spočtena pro všechny jeho potomky, dostává se program do závěrečné části výpočtu.

Výpočet je proveden jedním ze tří vzorců pro výpočet pravděpodobnosti bezporuchového provozu. O jaký vzorec se bude jednat, rozhoduje typ hradla, umístěné bezpodmínečně za aktuálně počítanou událostí. Na výběr jsou vzorce pro sériové zapojení (9) reprezentované hradlem OR, vzorec pro paralelní zapojení (11), které je zastoupené pomocí hradla AND nebo vzorcem (13), který reprezentuje hradlo K/N.

Po provedení výpočtu všech hodnot pravděpodobnosti bezporuchového provozu u daného objektu, se program přesune opět zpátky na začátek k testování vrcholové události a algoritmus probíhá znovu, dokud nejsou spočteny hodnoty pravděpodobnosti pro vrcholovou událost.

Pomocí shodného principu probíhají výpočty i při stanovení *F(t)* u neopravovaného systému, nebo *A(t), U(t)* u opravovaného systému.

3.11 Ukládání a načítání projektu

Programovací jazyky i jejich knihovny rozlišují mezi dvěma typy souborů. Jedná se o textový a binární soubor. O způsobu zápisu, rozhodneme při otevírání souboru.

Výhodou textových souborů je to, že jsou zapisovány v ASCII znacích a jsou tedy pro uživatele čitelné. Nevýhodou by mohlo být kódování souboru. Jelikož znaky jsou zapisovány v nějakém kódování, např. Windows-1250, UTF-8 apod. Při čtení souboru poté dochází k dekódování.

U binárního souboru jsou znaky zapisovány jako čísla v binární soustavě. Data se ukládají ve tvaru, v jakém jsou uloženy v paměti. Výhodou oproti textovým souborům je to, že zde nedochází k žádným konverzím. Další výhodou je utajení uložených dat. Při čtení binárního souboru musíme přesně vědět, jaký datový se bude právě načítat.

Při ukládání dat, ať už v textové či binární podobě, je nutné nejdříve otevřít proud (stream), který bude se souborem pracovat. Stream si můžeme představit jako nástroj pro přenos dat mezi programem a souborem. Po ukončení práce se souborem by se měl stream uzavřít.

Ukládání:

V práci je zvoleno ukládání dat pomocí binárních souborů. Program ukládá informace o všech událostech a hradlech. O jakou událost nebo hradlo se jedná (slovník $udalosti$, $poleHradel$), jakých hodnot dané události a hradla nabývají ($poleHodnot$) nebo o vztazích mezi jednotlivými událostmi a hradly ($potomci$).

Jak již bylo zmíněno v této kapitole, je nejdříve potřeba otevřít proud pro práci s binárním souborem. Pro to slouží třída *FileStream*, jejíž instanci získáme pomocí metody *Create*, kde parametrem bude cesta k cílovému binárnímu souboru. [11]

```
FileStream file = File.Create(„cesta k souboru“);
```

Dále je nutné otevřít další proud, který nám zajistí zápis do binárního souboru. Tento proud se nazývá *BinaryWriter*.

```
BinaryWriter writer = new BinaryWriter(file);
```

Po této inicializaci přejde program k samotnému ukládání dat. Pro zápis dat do souboru využívá program metodu třídy *BinaryWriter Write()*.

Před každým zápisem jednotlivého slovníku, zapíše program jako první velikost daného slovníku. Jako první program ukládá informace ze slovníku $poleHodnot$. Nejprve program zjistí, zda jsou již události vypočteny. To se zjistí ze slovníku $poleVypocteno$. Tato hodnota se uloží jako první. Jestliže je hodnota v tomto slovníku *false*, zapíše se počet událostí, pomocí:

```
writer.Write(poleHodnot.Count);
```

poté program prochází celý slovník $poleHodnot$ a zapisuje do souboru jednotlivé klíče slovníku, které reprezentují jednotlivé události a hradla. V opačném případě, tj. když je hodnota ve slovníku $poleVypocteno$ rovna *true*, zapíše program do souboru počet všech událostí a počet všech hodnot, které se na dané pozici nacházejí. Poté program prochází jednotlivé události a hradla a ukládá postupně *id* události a jednotlivé hodnoty (klíč a hodnota), které se na daném klíči nacházejí.

Dalším slovníkem, který program zapisuje je slovník $poleVypocteno$. Z tohoto slovníku program zapíše vždy *id* dané události a hodnotu *true* nebo *false*.

Jako další slovník, který se bude zapisovat je slovník $poleVstupnichHodnot$. Program zde zapíše do souboru *id* základní události a poté hodnoty, kterých událost nabývá.

Dalším slovníkem v pořadí je slovník $potomci$. Tento slovník je popsán v kapitole (3.8), který v sobě uchovává informace o vztazích mezi jednotlivými událostmi a hradly. Program nejdříve uloží klíč slovníku, který reprezentuje *id* jednotlivé události. Dále je uložena velikost datové struktury *List*, který se nachází na daném klíči a poté jednotlivé hodnoty v *Listu*, které reprezentují potomky dané události.

Jako poslední ukládá program informace ze zbývajících slovníků. Nejdříve slovník $poleHradel$ a poté slovník $udalosti$. Zapisování dat probíhá stejně jako u slovníku $poleVypocteno$. Nejdříve program uloží klíč a poté hodnoty, které se na daném klíči vyskytují.

Poté je celý cyklus zápisu dat dokončen a jednotlivé proudy se ukončí pomocí metody *Close()*.

Načítání:

Jak již bylo zmíněno, načítání dat z binárního souboru můžeme pouze v případě, že známe datový typ právě načítaných dat. Po otevření proudu se tedy postupně načítají všechna data, jak byla uložena při zapisování.

První hodnota, která se načte, určuje, zda byla vrcholová událost při zápisu vypočtena. V případě, že hodnota je *false*, načtou se do slovníku $poleHodnot$ pouze klíče. Je-li hodnota *true*, vrcholová událost byla již vypočtena a program načte klíč a dále k němu všechny hodnoty, které slovník obsahuje (3.5).

Dalším slovníkem, do kterého se budou načítat data, je slovník $poleVypocteno$. Jako první se načte klíč dané události, k němu poté hodnota *true* nebo *false*.

Stejným způsobem se načítají zbylé slovníky $poleVstupnichHodnot$, $potomci$, $poleHradel$ a $udalosti$.

3.12 Ukládání a načítání pracovní plochy

Aby bylo možné pokračovat v práci na načteném projektu, je nezbytné při ukládání projektu ukládat grafickou podobu stromu poruch. V programu je možné ukládat strom poruch buďto jako obrázek nebo stejně jako data ze slovníků do binárního souboru. Rozdíly mezi těmito soubory jsou stejné jako rozdíly mezi textovým a binárním souborem. Tudíž oproti obrázku nebude binární soubor, do kterého se uloží pracovní plocha, pro uživatele čitelný.

Pro ukládání pracovní plochy do obrázku se využívá metody *Save*. Tato metoda uloží daný objekt *Image* do souboru, který se udává jako parametr metody *Save*.

V případě ukládání objektu *Image* do binárního souboru, se stejně jako u ukládání dat ze slovníku nejprve otevře proud pro práci s binárním souborem. Pro ukládání objektu typu *Image* se využívá třídy *BinaryFormatter* a její metody *Serialize*, která serializuje objekt. Při ukládání pracovní plochy se ukládá *id* posledního vloženého objektu, z důvodu, aby při načtení uloženého projektu, mohl uživatel dále pokračovat v tvorbě stromu poruch [11] .

Serializace:

Jedná se o uchování stavu objektu. Daný objekt, který je serializovatelný, se převede na proud bytů a poté je uložen např. do databáze nebo do souboru.

Při načítání se nejdříve opět otevře proud pro práci s binárním souborem. Jelikož se při ukládání využívalo serializace objektu, využívá se při načítání deserializace. Ta je reprezentována metodou třídy *BinaryFormatter Deserialize*. Po načtení uloženého obrazu pracovní plochy načte program i uloženou hodnotu *id*, aby bylo možné pokračovat v tvorbě stromu poruch.

Deserializace:

Jedná se o opak serializace, kde se naopak proud bytů převede zpět na objekt.

3.13 Výstup programu

Po provedení výpočtu všech hodnot, které si uživatel zadá, má možnost zobrazit výsledky zvolené události buďto v grafické formě nebo v textové.

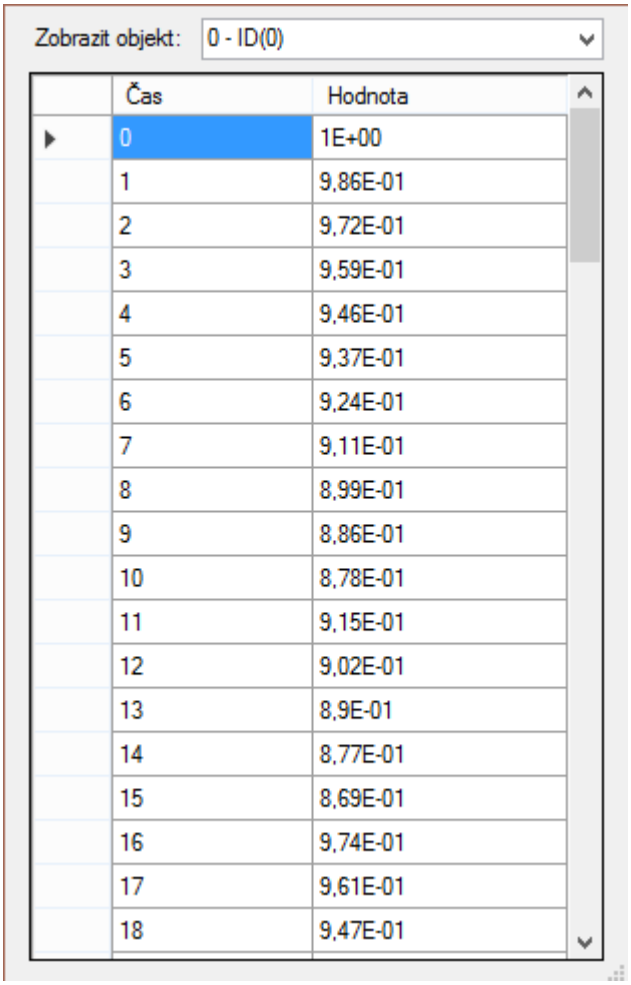
Uživatel si může zobrazit výsledná data jak pro vrcholovou událost, tak i pro všechny ostatní objekty. Jelikož se počítají výsledné hodnoty pro všechny objekty (slovník $poleHodnot$), stačí vždy vybrat daný objekt, u kterého chce uživatel zobrazit finální hodnoty výpočtu.

Pro další práci s výslednými hodnotami, si může uživatel tato data uložit do souboru ve formátu CSV. Data se ukládají postupně řádek po řádku. Data mají v souboru následující formát:

$$id\ objektu; čas; vypočtená\ hodnota\ v\ daném\ čase$$

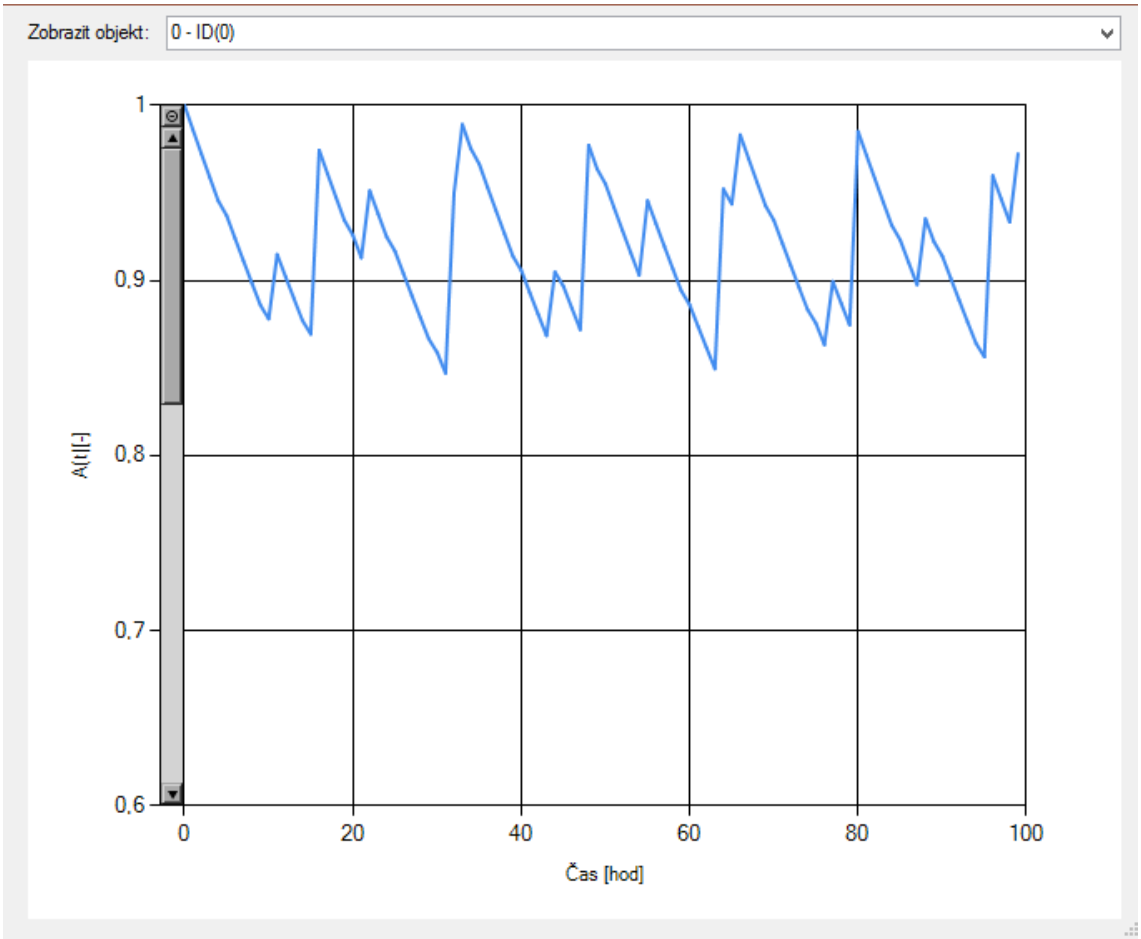
Jednotlivá data jsou oddělena středníkem, z důvodu výskytu desetinné čárky u vypočtených hodnot.

Textový výstup je reprezentovaný tabulkou, ve které první sloupec uchovává hodnoty času *t*, a ve druhém sloupci jsou obsaženy počítané finální hodnoty, které si při zadávání parametrů výpočtu zvolil (Obrázek 15). Uživatel si může vybírat jednotlivé výsledné hodnoty daných objektů přes *combo box*, který se nachází v horní části okna. Příklad této tabulky hodnot je zobrazen na obrázku (Obrázek 17).



Obrázek 17: Příklad textového výstupu programu

Druhou možností zobrazení dat je grafická reprezentace. Jedná se o spojnicový graf, kde na ose *x* jsou zaznamenány hodnoty času (*t*), jednotky, ve kterých je čas zobrazen si uživatel vybírá při nastavení parametrů výpočtu (Obrázek 15). Na ose *y* poté vypočítané finální hodnoty dané události, které stejně jako čas a jednotky výpočtu vybírá uživatel při nastavení parametrů pro výpočet. Stejně jako u zobrazení dat v tabulkové podobě, i zde si může uživatel zobrazit grafy jednotlivých objektů. Příklad grafu je zobrazen na obrázku (Obrázek 18).

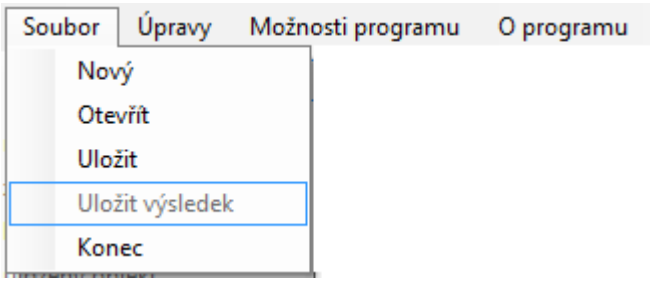


Obrázek 18: Příklad grafického výstupu programu

3.14 Dokumentace

Na obrázku (Příloha 2) je vidět úvodní obrazovka, která uživateli naběhne ihned po spuštění programu. Hlavní lišta nabízí několik možností. Karta *Soubor* nabízí několik možností práce s projektem (Obrázek 19). Tyto možnosti jsou:

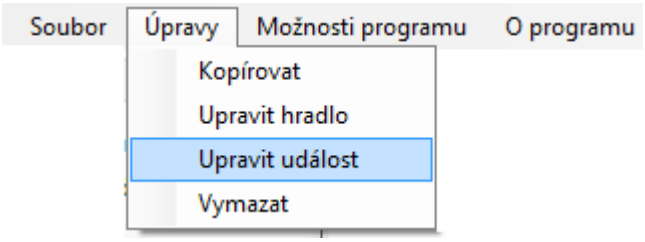
- Nový – nastaví program do základního nastavení, jako při spuštění programu
- Otevřít – nahraje všechna data z binárního souboru
- Uložit – uloží dosavadní postup do binárního souboru
- Uložit výsledné hodnoty – uloží výsledky do souboru CSV
- Konec – ukončí program



Obrázek 19: Přehled možností na kartě Soubor

Další kartou jsou *Úpravy*. Tato karta obsahuje možnosti úpravy a nastavení základních událostí a hradel. (Obrázek 20)

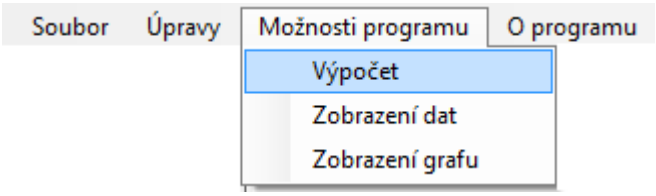
- Kopírovat – zkopíruje vybraný objekt
- Upravit hradlo – otevře okno pro nastavení parametrů zvoleného hradla
- Upravit událost – otevře okno pro nastavení vstupních parametrů u vybrané základní události
- Vymazat – vymaže označené hradlo nebo událost a vymaže také všechny jeho potomky



Obrázek 20: Přehled možností na kartě Úpravy

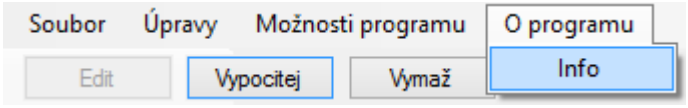
Předposlední kartou je karta *Možnosti programu*. Na této kartě se nachází možnosti od výpočtu programu až do zobrazení výsledků (Obrázek 21)

- Výpočet – otevře okno pro nastavení parametrů finálního výpočtu (obrázek…) a po nastavení těchto parametrů proběhne výpočet
- Zobrazení grafu – otevře okno s grafickým zobrazením výsledků výpočtu zvolené události nebo hradla
- Zobrazení dat – otevře okno s tabulkovou reprezentací vypočtených dat zvolené události nebo hradla



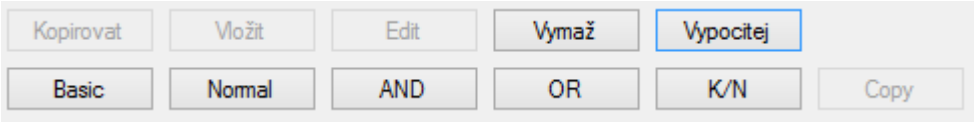
Obrázek 21: Přehled možností na kartě Možnosti programu

Poslední karta obsahuje informace o programu (Obrázek 22).



Obrázek 22: Přehled možností na kartě O programu

Další částí programu je nabídka všech objektů, které může uživatel vložit (Obrázek 23). Jedná se o základní událost a normální událost, které jsou více popsány v kapitolách (1.4.2 a 1.4.3). Dále hradlo. Toto hradlo je univerzální a až při jeho editaci uživatel zvolí, o jaké hradlo půjde. Je na výběr ze tří možností, OR, AND a K/N (Obrázek 10). Poslední položkou, kterou může uživatel přidat na pracovní plochu je předem zkopírovaný objekt. Objekt si také uchovává nastavené hodnoty zkopírovaného objektu.

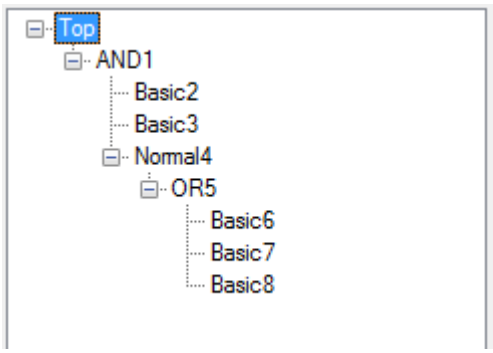


Obrázek 23: Nabídka dostupných objektů

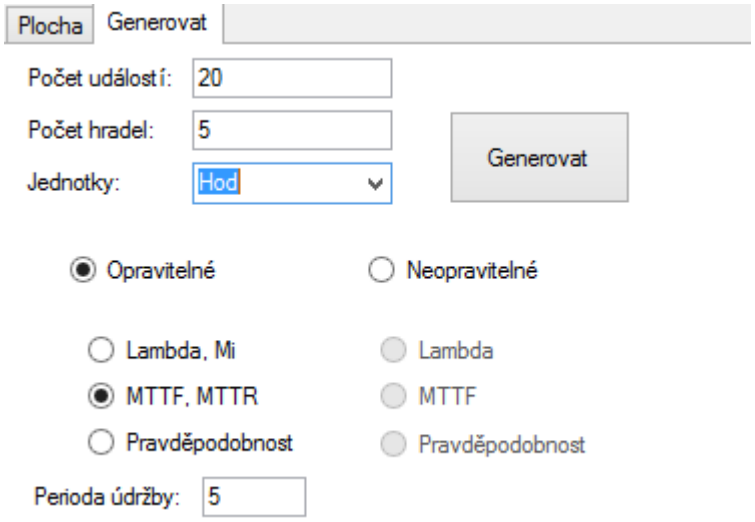
Tvorba stromu

Uživatel má několik možností tvorby stromu poruch. První možností je vytvářet strom poruch přímo na pracovní ploše. Označit objekt, který bude předkem vkládaného objektu. Dále zvolit, jaký objekt bude vložen a následně kliknout na pracovní plochu na místo, kde se má objekt zobrazit (Příloha 3).

Druhou možností je vkládání objektu do textového zobrazení stromu (*Treeview*). Uživatel označí objekt, který bude předkem vkládaného objektu a klikne na objekt, který chce vložit. [11] [12]

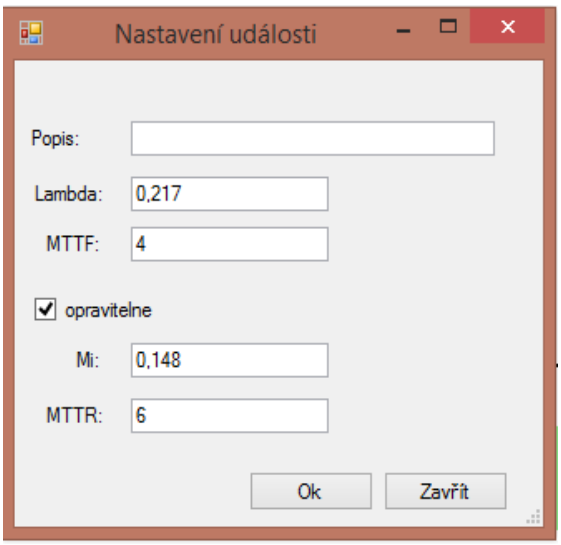


Poslední možností je, si celý strom poruch náhodně vygenerovat. Uživatel si přepne záložku z pracovní plochy na generátor. Zde vyplní počet událostí, počet hradel, které bude daný strom obsahovat. Dále vyplní jednotky a zvolí, zda se bude jednat o opravitelný či neopravitelný systém. V případě, že si zvolí opravitelný systém, přibude k vyplněným parametrům ještě parametr *perioda obnovy*.



Obrázek 24: Nastavení parametrů pro náhodné generování stromu

Nastavení parametrů základních událostí:



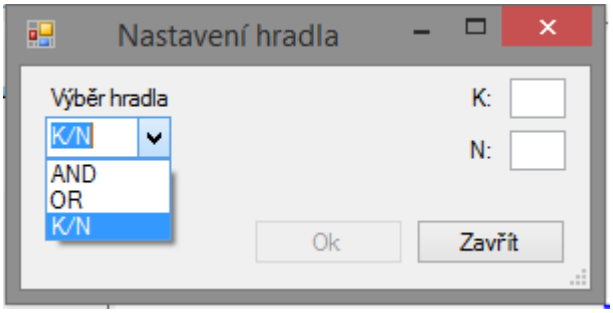
U základní události nastavujeme vstupní parametry pro výpočet λ, μ, $MTTR$ a $MTTF$. Všechny tyto parametry jsou popsány blíže v kapitole (1.5.1). Ještě než začneme zadávat hodnoty do příslušných polí, musíme vybrat, jakou dvojici vstupních hodnot budeme zadávat. Jak je vidět na obrázku (Obrázek 9), jedná se o dvojice λ, μ a $MTTR$, $MTTF$. Dalšími parametry jsou *název* a *popis*. Tyto pole slouží pro popis dané události.

Pokud uživatel zvolí možnost textové tvorby nebo generování stromu poruch, může dané parametry vyplnit nebo změnit v tabulce hodnot, která se zobrazí v pravé části okna. [11] [12]

	Name	Value	Unit
	Popis		-
	Lambda	0,001	Hod^-1
	Mi	0,002	Hod^-1
	MTTR	500	Hod
	MTTF	1000	Hod
	Perioda údržby	10	Hod
▶	Vysledek	0	Hod
*			

Nastavení parametrů hradel:

U tohoto objektu vybíráme ze tří dostupných typů hradel (Obrázek 10). Jedná se o hradla typu OR, AND a K/N (1.4.5). [11] [12]

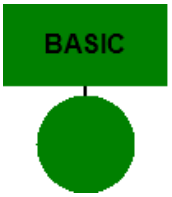


Pokud uživatel zvolí možnost textové tvorby nebo generování stromu poruch, může dané parametry vyplnit nebo změnit v tabulce hodnot, která se zobrazí v pravé části okna.

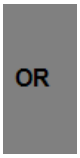
Zde ukázka nastavení hradla *K/N*.

	Name	Value	Unit
	Popis		-
	K	2	-
▶	N	4	-
	Vysledek	0	
*			

Přehled dostupných objektů:

Název události	Popis	Značka
TOP	Vrcholová událost	
NORMAL	Normální událost dále rozvíjená	
BASIC	Základní událost Koncová událost, není již dále rozvíjena	

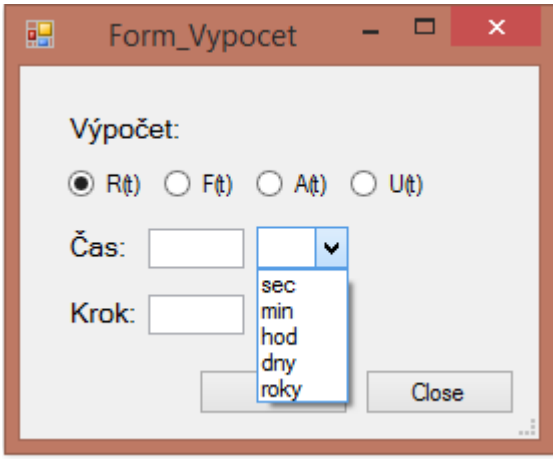
Tabulka 2: Přehled událostí používaných v programu

Název hradla	Vlastnosti	Značka
OR	Reprezentuje sériové zapojení systému	
AND	Reprezentuje paralelní zapojení systému	
K/N	Reprezentuje zapojení pomocí hradla K prvků z N možných	

Tabulka 3: Přehled používaných hradel v programu

Výpočet:

Po nastavení všech vstupních informací (parametry u základních událostí a hradel) můžeme daný systém vypočítat. Pro výpočet slouží záložka v hlavním panelu na kartě *Možnosti programu* nebo tlačítko *Výpočet* pod hlavním panelem. Po kliknutí na výpočet se zobrazí okno pro nastavení parametrů výpočtu.



Zde je několik parametrů, které musí uživatel nastavit.

- Typ výpočtu – na výběr je z možností $R(t)$, $F(t)$, $A(t)$ a $\mathrm{U}(t)$
- Simulační čas
- Jednotky simulačního času (sek, min, hod, …) [12]
- Krok – po jakých krocích z celkového simulačního času se má výpočet provádět

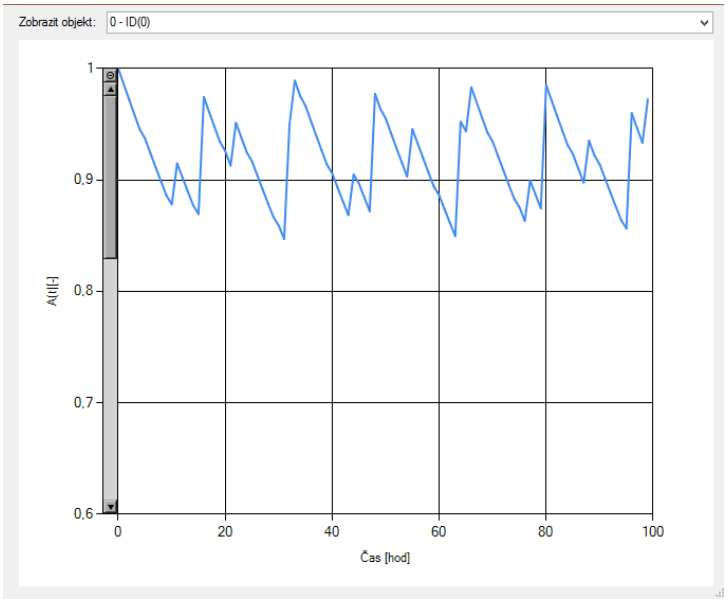
Zobrazení výsledků:

Po skončení výpočtu, si může uživatel vypočtené hodnoty zobrazit. Na výběr má ze dvou možností:

- Pomocí grafu
- Pomocí tabulky

Při zvolení grafického výstupu, se otevře okno se spojnicovým grafem, který reprezentuje jednotlivé výsledky v daném čase. V horní části okna s výsledným grafem se nachází *combo* pro výběr objektu. Pomocí něj může uživatel vybírat a zobrazovat výsledné grafy jednotlivých objektů. [12]

Při výběru textového výstupu, se otevře okno s tabulkou finálních hodnot. Tabulka obsahuje dva sloupce, kde první reprezentuje jednotlivé časové kroky a druhý obsahuje výsledné hodnoty daných časových kroků. Stejně jako u grafického zobrazení výstupu, i zde je možné si pomocí *comba* vybírat mezi jednotlivými objekty.



Zobrazit objekt: 0 - ID(0)

Čas	Hodnota
0	1E+00
1	9,86E-01
2	9,72E-01
3	9,59E-01
4	9,46E-01
5	9,37E-01
6	9,24E-01
7	9,11E-01
8	8,99E-01
9	8,86E-01
10	8,78E-01
11	9,15E-01
12	9,02E-01
13	8,9E-01
14	8,77E-01
15	8,69E-01
16	9,74E-01
17	9,61E-01
18	9,47E-01

Ukládání výsledků:

Další funkcionalita, které se zpřístupní po vykonání výpočtu, je ukládání vypočtených hodnot. Aby bylo možné s vypočtenými daty dále pracovat, ukládají se tyto hodnoty do souboru typu CSV. Data se ukládají postupně řádek po řádku. Každý řádek obsahuje *id* objektu, časovou hodnotu a vypočtenou hodnotu v daném čase.

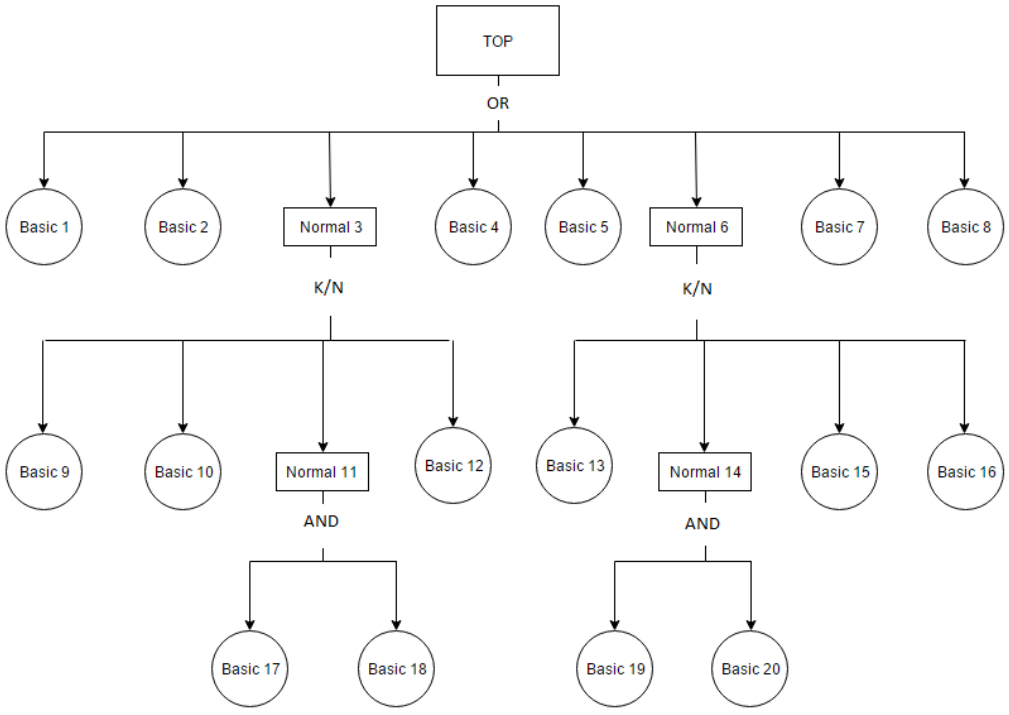
Jednotlivá data jsou oddělena středníkem, z důvodu výskytu desetinné čárky ve vypočtených hodnotách.

```
0;87;0,00226540891481432
0;90;0,00183630477702891
0;93;0,00148847972305943
0;96;0,00120653821395804
0;99;0,000978000868395394
```

Obrázek 25: Ukázka uložených dat u vrcholové události v souboru CSV

3.15 Testovací příklad

Pro testování funkčnosti a správnosti výsledků byl zvolen následující strom poruch (Obrázek 26). Strom obsahuje všechna dostupná hradla. Nachází se zde hradla typu AND, OR i K/N.



Obrázek 26: Testovací příklad

Pro testovací strom byly použity následující vstupní hodnoty pro základní události. (Tabulka 4)

Číslo základní události	Vstupní hodnoty (MTTF)
1	10000
2	10000
4	10000
5	10000
7	10000
8	10000
9	1000
10	1000
12	1000
13	1000
15	1000
16	1000
17	500
18	500
19	500
20	500

Tabulka 4: Přehled vstupních hodnot v testovacím příkladu

Pro výpočet byly zvoleny následující parametry:

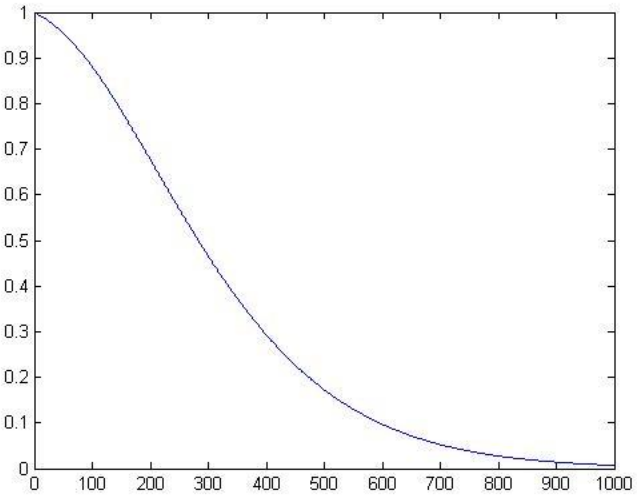
- Typ výpočtu – $R(t)$
- Simulační čas – 1000 hodin
- Krok výpočtu – 10 hodin

Výsledné hodnoty programu byly porovnány s analytickým řešením. Oba výsledky jsou zobrazeny v textové i grafické podobě.

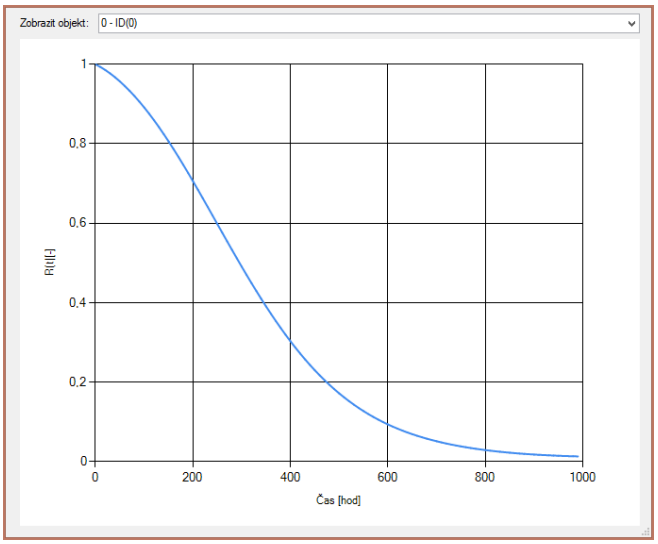
Čas	Analytické řešení	Programové řešení
0	1	1
50	0,955	0,957
100	0,88	0,891
150	0,784	0,806
200	0,677	0,706
250	0,568	0,598
300	0,465	0,491
350	0,373	0,391
400	0,294	0,303
450	0,23	0,23
500	0,173	0,172
550	0,131	0,127
600	0,098	0,094
650	0,0721	0,0694
700	0,0528	0,0516
750	0,0384	0,0387

800	0,0278	0,0294
850	0,0199	0,0228
900	0,0143	0,0182
950	0,0102	0,0151

Tabulka 5: Přehled výsledků analytického a programového řešení



Obrázek 27: Grafický výstup analytického řešení



Obrázek 28: Grafický výstup programového řešení

Závěr

Cílem diplomové práce bylo nastudování problematiky metody FTA, otestování a rešerše dosavadních softwarů a vytvoření vlastního programu. Základní popis metody je uveden v kapitole 1, pro detailnější studium lze využít normy (1), manuálů ke komerčním SW (viz kapitola 2).

Druhým bodem bylo otestování některých komerčních softwarů, které metodu FTA počítají. Protože většina softwarů je placená, byly využity trial verze. U každého softwaru bylo testováno, které základní hradla program využívá, jak program sestavuje strom poruch, nastavení vstupních proměnných, zobrazení výstupních dat.

Posledním bodem práce bylo vytvoření aplikace, která bude počítat spolehlivost zadaného stromu poruch pomocí metody FTA. Aplikace pro tvorbu stromu poruch využívá základní logické hradla (OR, AND a K/N) a tři druhy událostí (základní, normální a vrcholovou). Uživatel si vytváří strom poruch pomocí těchto objektů a zadává jim vstupní parametry. Další možností jak si může uživatel vytvořit strom poruch je náhodné generování stromu. Při generování se vyplňují počty hradel a událostí, které bude vytvořený strom poruch obsahovat. Aplikace počítá hodnoty $R(t)$, $F(t)$, $\mathrm{A}(t)$ a $\mathrm{U}(t)$ dle zadaných vstupních parametrů. Výsledky si může uživatel zobrazit jak ve formě tabulky, tak i ve formě grafu. Výsledné hodnoty může také uložit do souboru typu CSV. Práce obsahuje testovací příklad, který je zobrazen na obrázku č. Obrázek 26 a dokumentaci pro uživatele (viz kapitolu 3.14)

Zdroje

[1] ČSN EN 61025 (010676). *Analýza stromu poruchových stavů*. 2007.

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

[3] ČSN IEC 1025 (010676). *Analýza stromu poruchových stavov*. 1994.

[4] ERICSON II, Clifton A. *Fault Tree Analysis* [online]. In: . [cit. 2016-05-20]. Dostupné z: http://www.thecourse-pm.com/Library/FaultTreeAnalysis2.pdf

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

[6] *ALD Profile - ALD Service* [online]. 2016 [cit. 2016-05-20]. Dostupné z: http://aldservice.com/Company/ald-profile.html

[7] *FTA (Fault Tree Analysis) - Analýza stromu poruchových stavů* [online]. 2015 [cit. 2016-05-20]. Dostupné z: https://managementmania.com/cs/fault-tree-analysis

[8] *ETA (Event tree analysis) - analýza stromu událostí* [online]. 2015 [cit. 2016-05-20]. Dostupné z: https://managementmania.com/cs/eta-event-tree-analysis-analyza-stromu-udalosti

[9] *FMEA (Failure Mode and Effect Analysis)* [online]. 2016 [cit. 2016-05-20]. Dostupné z: https://managementmania.com/cs/failure-mode-and-effect-analysis

[10] *Fault Tree Analysis (FTA, System Analysis) Basics* [online]. 2016 [cit. 2016-05-20]. Dostupné z: http://www.weibull.com/basics/fault-tree/

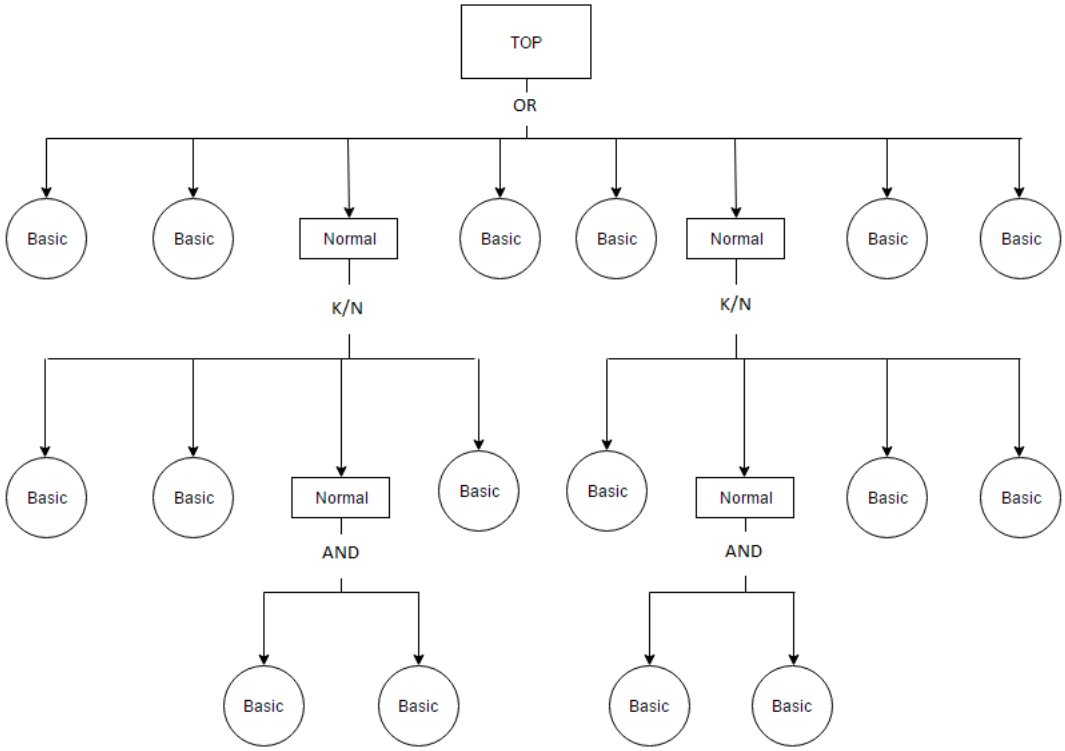
[11] MAREŠ, Amadeo. *1001 tipů a triků pro C# 2010*. Brno: Computer Press, 2011. ISBN 978-80-251-3250-0.

[12] PETZOLD, Charles. *Programování Microsoft Windows Forms v jazyce C#: [vytváříme uživatelské rozhraní aplikací]*. Brno: Computer Press, 2006. ISBN 80-251-1058-3.

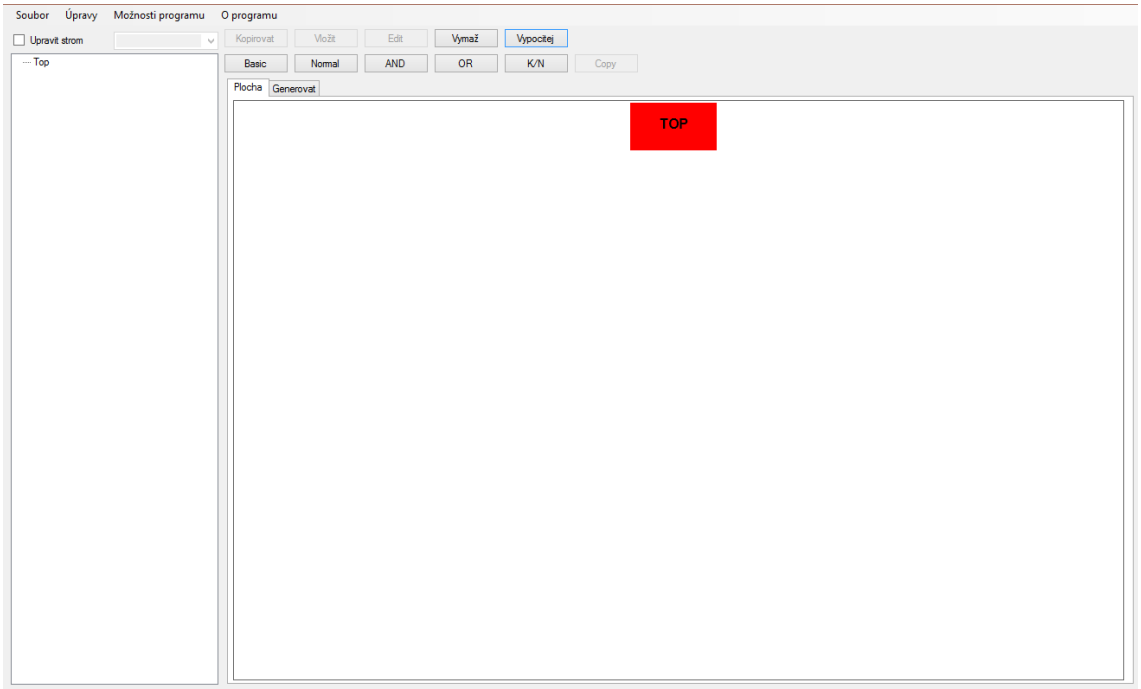
[13] ČADA, Ondřej. *Objektové programování: naučte se pravidla objektového myšlení*. Praha: Grada, 2009. Průvodce (Grada). ISBN 978-80-247-2745-5.

[14] KEOGH, James Edward a Mario GIANNINI. *OOP bez předchozích znalostí: průvodce pro samouky*. Brno: Computer Press, 2006. ISBN 80-251-0973-9.

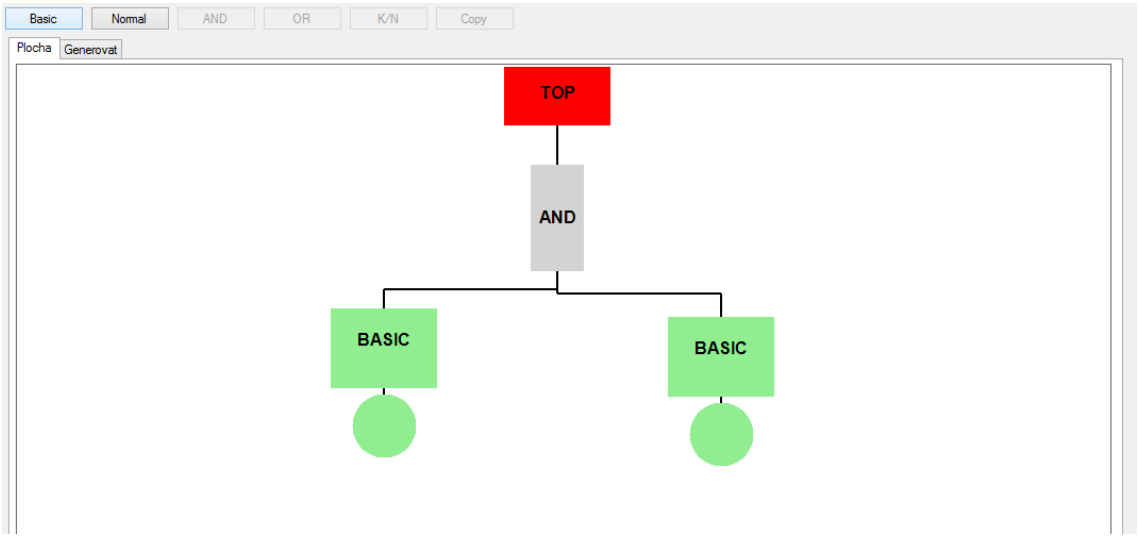
Přílohy



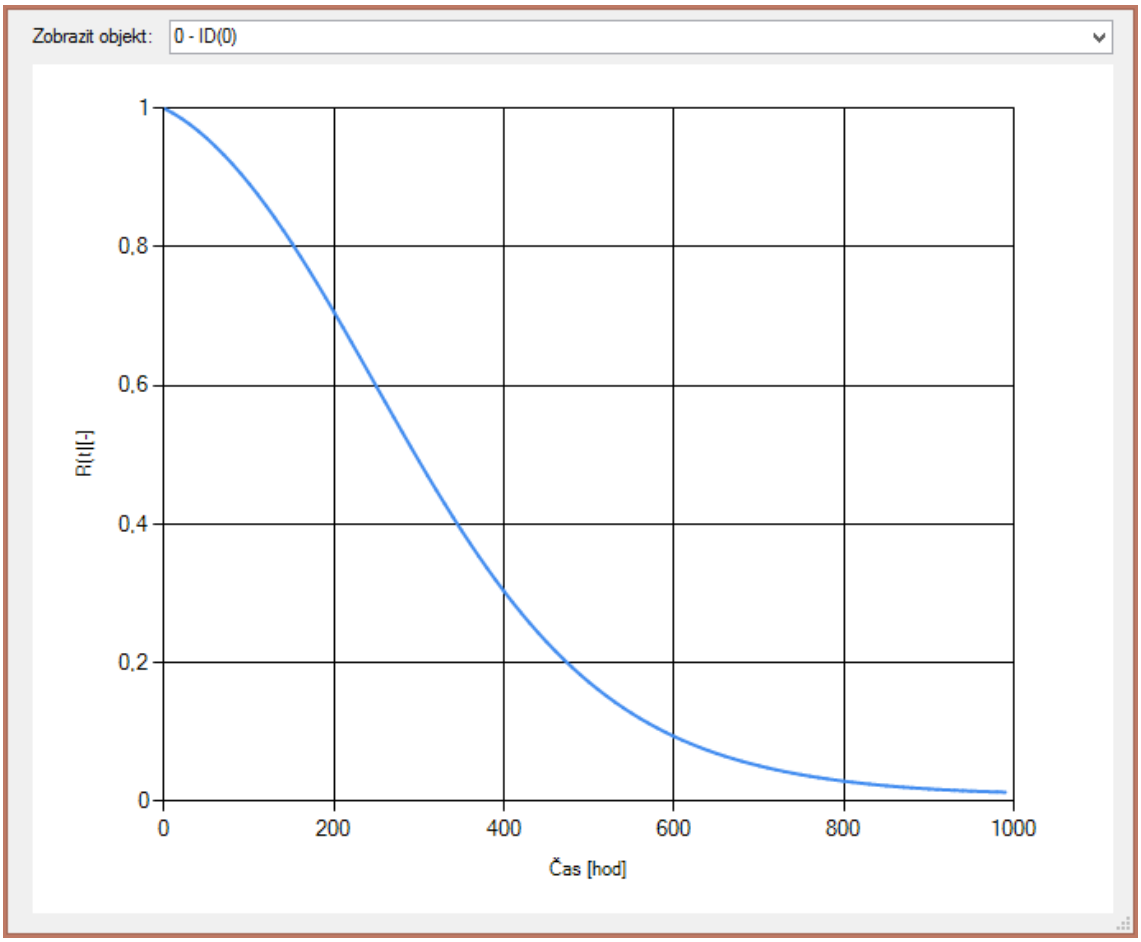
Příloha 1: Ukázka testovacího příkladu



Příloha 2: Ukázka úvodní obrazovky programu



Příloha 3: Příklad tvorby stromu pomocí grafického rozhraní



Příloha 4: Výsledný graf vrcholové události testovacího příkladu