VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

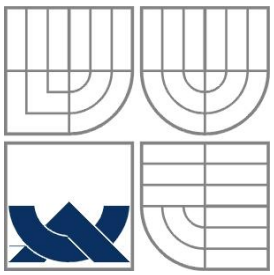
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZHODOVACÍ METODY V MANAGEMENTU RIZIK

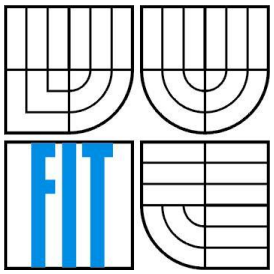
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PETR JANOŠÍK
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZHODOVACÍ METODY V MANAGEMENTU RIZIK

DECISION RISKS MANAGEMENT METHODS

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE — Bc. PETR JANOŠÍK
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. JITKA KRESLÍKOVÁ, CSc.
SUPERVISOR

BRNO 2011

Abstrakt

Tato diplomová práce se zabývá problematikou managamentu rizik v projektech IT. Vysvětluje důležitost řízení rizik v projektech a ukazuje možné postupy a metody, jak rizika řídit a analyzovat. Po vysvětlení základních pojmů a jednotlivých fází řízení rizik je práce zaměřena na dvě metody analýzy rizik, a to na analýzu stromu chyb a analýzu stromu událostí. Je zde vysvětlen postup použití obou metod jak pro kvantitativní, tak pro kvalitativní analýzu. V druhé polovině práce je uveden návrh aplikace pro podporu analýzy rizik za pomoci analýzy stromu chyb a stromu událostí, na který navazuje popis implementace navrženého systému ve webovém prostředí s využitím nástrojů jQuery, Nette Framework a Dibi.

Abstract

This thesis deals with the matter of risk managament in IT projects. It explains the importance of risk management in such projects and shows different ways and methods of managing and analyzing the risks. After explaining the basic concepts and the various phases of risk management the text focuses on two methods of risk analysis – the fault tree analysis of event tree analysis. Use of both methods is explained for both quantitative and qualitative analyses. The second half of the work includes the design of an application for the support of risk analysis employing the methods of fault tree analysis and event tree analysis. This is followed by a description of the implementation of the proposed system in a web environment using jQuery, Nette Framework and Dibi.

Klíčová slova

riziko, management rizik, projekt, aktiva, hrozby, zranitelnost, dopad, ochranná opatření, zbytkové riziko, analýza rizik, strom chyb, strom událostí, identifikace rizik, kvantitativní analýza, kvalitativní analýza, monitoring rizik, MVP, Nette Framework

Keywords

risk, risk management, project, asset, threat, vulnerability, impact, protective measures, residual risk, risk analysis, fault tree , event tree, risk identification, quantitative analysis, qualitative analysis, risk monitoring, MVP, Nette Framework

Citace

Janošík Petr: Rozhodovací metody v managementu rizik, diplomová práce, Brno, FIT VUT v Brně, 2011

Rozhodovací metody v managementu rizik

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. RNDr. Jitky Kreslíkové, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Janošík
25.5.2011

Poděkování

Děkuji doc. RNDr. Jitce Kreslíkové, CSc za odborné vedení, cenné rady, konzultace a připomínky, které přispěly k dokončení mé diplomové práce. Rovněž děkuji všem, kteří mě při této práci podporovali a pomáhali.

Obsah

1 Úvod

Již několik let se velké společnosti snaží překonat ekonomickou krizi, která stále ještě neutichla. Mnoho společností muselo ukončit svoji působnost na trhu, protože neunesly těžké podmínky, které nastaly. Mohly se některé společnosti lépe připravit na rizika, která je postihla? Z výzkumů vyplývá, že v procesu řízení projektů v oblasti průmyslu vývoje softwaru je nejnižší stupeň vyspělosti právě v řízení rizik [4]. Tato situace může být způsobena tím, že řízení rizik je jedním z často přehlížených aspektů při řízení projektů, nebo tím, že organizace na řízení rizik nevyčlení dostatek financí a sázejí na to, že projekt dokončí i bez důsledného managamentu rizik.

Tato práce shromažďuje informace, které by měl každý projektový manažer znát. Na začátku druhé kapitoly jsou popsány základní pojmy, mezi které patří projekt, aktiva, hrozba, zranitelnost, apod. V další části kapitoly je nastíněn úvod do řízení a plánování rizik, kde jsou mimo jiné popsány různé statistiky spojené s řízením rizik a vnímání rizik organizacemi. Také jsou popsány nejčastější zdroje rizik v projektech IT a je vysvětleno rozdělení rizik do skupin. Po této části se zmíním o identifikaci rizik, technikách používaných k identifikaci a způsobech zapisování identifikovaných rizik. Dále je popsána kvantitativní a kvalitativní analýza a poté plánování reakcí na analyzovaná rizika. Na konci kapitoly je dán důraz na monitorování a kontrolu rizik.

Třetí kapitola se zabývá technikami v managementu rizik ve fázi kvalitativní a kvantitativní analýzy rizik. Zaměřil jsem se na analýzu stromu chyb a analýzu stromu událostí. V obou technikách je popsán postup použití a vysvětleno, kde a kdy jakou metodu použít. V závěru této kapitoly jsou kratší a jednodušší formou popsány ještě některé další techniky.

Čtvrtá kapitola je zaměřena na analýzu a návrh aplikace pro podporu kvalitativní analýzy rizik za pomocí analýzy stromu chyb a analýzy stromu událostí. Je zde popsána neformální specifikace aplikace, zakreslen diagram případu užití, návrh databáze a navrženo předběžné rozmístění grafických prvků.

Z analýzy a návrhu systému vychází pátá kapitola, popisující implementaci navrženého systému. Na začátku kapitoly jsou popsány použité programovací jazyky a nástroje, mezi které patří jQuery, Nette Framework, Dibi atd. Z velké části je zde popsán Nette Framework, který je jádrem celého systému. Je zde vysvětleno, co je softwarová architektura MVC, a rozdíl v porovnání s architekturou MVP, kterou systém využívá. V neposlední řadě tato kapitola vysvětluje složitost implementace výpočtu frekvence a pravděpodobnosti analýzy stromu chyb a stromu událostí.

Šestá kapitola shrnuje všechny funkce systému a zároveň vysvětluje jejich použití. Jsou zde popsány výhody využití právě tohoto implementovaného systému. Na tuto kapitolu poté navazuje předposlední kapitola ověřující správnost implementace algoritmu počítajícího analýzy stromu chyb a stromu událostí a ucelující zkušenosti získané v průběhu čtení této práce.

V poslední kapitole je práce shrnuta v několika odstavcích a jsou sepsány nové zkušenosti, které jsem díky tomuto projektu získal. Součástí shrnutí je také zhodnocení implementovaného systému. Jak každý ví, libovolný systém je možné neustále upravovat a rozšiřovat a i tento vytvořený systém není výjimkou, proto je v závěru této práce zahrnuta úvaha o možnostech dalšího rozšíření systému.

Tato práce navazuje na Semestrální projekt, který obsahoval část teoretického úvodu až po návrh aplikace. Velké části teoretického úvodu byly proto převzaty a po malých úpravách a doplnění informací použity i pro tuto práci.

2 Riziko a management rizik v projektech IT

Organizace si začínají uvědomovat, že management rizik je nedílnou součástí řízení projektů. Avšak je otázkou, kolik organizací ví, jak tuto činnost dělat nejefektivněji. Například jestli je vhodné mít rizika iniciována v průběhu celého životního cyklu projektu, nebo jen v některých jeho částech, popřípadě mít proces managementu rizik jako samotný hlavní cyklus činnosti.

V této kapitole rozebereme základní pojmy a definice, které je potřeba znát pro pochopení další části textu. Dále bude objasněno, co je to riziko, a proč je důležité mít v projektu dobré řízení rizik. Zjistíme, jak identifikovat rizika, a jak je následně analyzovat kvalitativně a kvantitativně. Na závěr kapitoly bude diskutováno o plánování reakcí na rizika a následné sledování a o kontrole definovaných rizik.

2.1 Vymezení základních pojmů

2.1.1 Projekt

Projekt je v managementu definován jako jedinečný proces, který je časově ohraničen a skládá se z řady činností za účelem dosažení předem stanoveného unikátního cíle. Projekt často bývá definován jako proces, což je chyba. V projektu je klíčové časové omezení a jedinečnost výstupu, čímž se odlišuje od procesu. Projektem tedy není činnost, u které není jasně definováno, kdy skončí, a jaký bude její výstup a především není činnost, která je realizována opakovaně. Projekt je vedle času a cíle také ohraničen zdroji, které jsou k dispozici pro jeho splnění. Z toho vyplývá, že jakákoliv změna jednoho z těchto ohraničení ovlivní změnu ostatních. Těmto třem veličinám se říká projektový trojimperativ [3].

2.1.2 Aktiva

Aktiva v oblasti IT je cokoliv, co má pro organizaci/společnost hodnotu. Pro úspěch organizace je důležité mít správné řízení aktiv. Řízení aktiv je hlavní odpovědností všech úrovní managementu. Mezi aktiva organizace patří:

- Fyzická aktiva (např. počítačový hardware, komunikační prostředky, budovy)
- Software
- Schopnost vytvářet určité produkty nebo poskytovat služby
- Lidé
- Nehmotné hodnoty (např. abstraktní hodnota firmy, informace, image)

Pokud nejsou aktiva chráněna, je zapotřebí udělat odhad rizik, který jsme ochotni akceptovat před případnou ztrátou nebo prolomením nedostatečné ochrany. Některé aktiva totiž můžou mít menší peněžní hodnotu, než investice vynaložené do prvotřídní ochrany nebo zabezpečení.

Abychom mohli aktiva nějak chránit, musíme je nejdříve identifikovat. Identifikace a následná analýza aktiv může vyžadovat, ale neplatí to vždy, poměrně časovou náročnost. Stupeň podrobností pro tuto analýzu je měřen poměrem času a nákladů vůči dané hodnotě aktiv. V mnoha případech jsou aktiva sdružena do skupin, což muže být užitečné.

Požadavky na ochranu aktiva vyplývají z jejich zranitelností při výskytu specifických hrozeb. Pro různé oblasti bývají rozdílné hrozby. Například v některých státech je ochrana osobních údajů velmi důležitá, zatímco jiné státy nekladou na toto opatření tak silný důraz.

2.1.3 Hrozby

Jak již bylo řečeno, každé aktivum je vystaveno určitým hrozbám, proti kterým se stanovují bezpečností opatření, protože hrozba má potenciální schopnost způsobit nežádoucí incident, který by mohl poškodit aktiva firmy. Aby hrozba mohla způsobit poškození aktiv, potřebuje k tomu zranitelná místa aktiv. Hrozba ale nemusí mít jen lidský původ ve smyslu „hackování“ systému. Hrozby můžou mít také přírodní původ a mohou být náhodné nebo úmyslné. Jak náhodné, tak úmyslné nežádoucí účinky by měly být identifikovány a měla by k nim být přiřazena pravděpodobnost vzniku.

Mezi úmyslné lidské hrozby například patří odposlech, krádež, hacking apod. Mezi náhodné lidské hrozby patří fyzické nehody, neúmyslné smazání souboru apod. a mezi hrozby prostředí neboli přírodní hrozby řadíme zemětřesení, blesk, požár apod. Velikost škody způsobená hrozbou může být za každé situace jiná. To znamená, že softwarový virus může způsobit různou škodu podle toho, jaké má naprogramované akce. Stejně tak může zemětřesení působit na konkrétním místě jinak, než v epicentru výskytu. Proto se k takovýmto hrozbám přiřazuje stupeň síly, která je s nimi spojená. Například virus může být destruktivní, nebo nedestruktivní a zemětřesení můžeme popsat pomocí Richterovi stupnice. Dále pak některé hrozby mohou působit na více aktiv najednou a u každé můžou mít jiný dopad. Záleží na tom, jaká aktiva jsou ovlivněna. Například pokud virus zaútočí na osobní počítač, tak hrozby nejsou tak velké, jako když stejný virus zaútočí na serverový počítač.

2.1.4 Zranitelnost

Zranitelnost aktiv jsou označována jako slabá místa na úrovni fyzické, organizační, procedurální, personální, řídící, administrativní, hardware, software nebo informací [1]. Zranitelnost sama o sobě není příčinou škody. Zranitelnost pouze dovoluje hrozbě, aby mohla ovlivnit aktiva. Například neřízený přístup do systému je zranitelnost, které by mohla využít hrozba nežádoucího proniknutí k aktivům (např. osobní informace klientů). Ale ne všechny zranitelnosti jsou lehce identifikovatelné a navíc záplatou některé ze slabých míst, může vzniknout nové slabé místo.

2.1.5 Dopad

Dopad je výsledek nežádoucího incidentu na aktiva. Tento incident byl způsoben buď úmyslně, nebo náhodně. Dopad může mít podobu poškození systému IT, ztráty důvěrnosti, integrity, dostupnosti nebo spolehlivosti. Nepřímé následky se odrazí na finanční ztrátu a z toho odvíjený zmenšený podíl na trhu nebo snížený image a prestiže společnosti. Měření dopadů se uplatňuje při určování nákladů na ochranná opatření před nežádoucími incidenty. Při určování nákladů nesmíme

zapomenout na zohlednění četnosti výskytu nežádoucích incidentů, protože u některých hrozeb může být velikost škody nízká, ale při častém výskytu může být souhrn těchto škod velký.

2.1.6 Riziko

Podle obvyklého slovníkového významu je riziko „vyhlídka na špatné časy“. V pojišťovnictví je riziko určeno jako matematické očekávání peněžní hodnoty, která je vyčíslena pomocí zapříčiněné nežádoucí událostí. V technických oborech je většinou riziko definováno jako jakákoliv možnost, která může způsobit škodu, jestliže existuje nejistota, zda tyto škody nastanou, nebo nenastanou [2].

Potenciální možnost rizika vzniká v případě, když daná hrozba využije zranitelnosti. Pokud je pravděpodobnost rizika velká, je také velká pravděpodobnost dopadu nežádoucího incidentu a tedy poškození aktiv respektive organizace. Rizika bývají charakterizována pravděpodobností výskytu nežádoucího incidentu a jeho dopadu. Pokud organizace nabude nová aktiva nebo vzniknou změny v hrozbách, zranitelnostech a ochranných opatření, musíme si uvědomit, že tyto nové aktiva a změny doprovází také změny v rizicích. Pro chod každé organizace je důležité včas identifikovat rizika a k nim připravit vhodné opatření ke snížení těchto rizik. V dalších kapitolách bude popsáno více o identifikaci, analýze a řízení rizik. A také zjistíme, že riziko nemusí mít vždy pouze negativní dopad.

2.1.7 Ochranná opatření

Ochranná opatření jsou postupy nebo mechanismy, které snižují počet slabých míst a tím pádem poskytují ochranu před hrozbou. Omezují dopad nežádoucího incidentu a můžou usnadnit obnovu škody. Pokud se snažíme vyvinout účinnou bezpečnost, tak se většinou jedná o kombinaci různých ochranných opatření, která poskytují různé stupně bezpečnosti. Mezi ochranná opatření patří například tyto funkce: detekce, omezení, obnova, monitorování, atd. Ochranné opatření je také pouhé podvědomí o bezpečnosti. Nesmíme ale zapomenout na to, že každé nové ochranné opatření v sobě může obsahovat nové slabé místo. Samotné příklady ochranných opatření jsou např. tyto:

- Síťové firewally a antivirové systémy
- Monitorování a analýza sítě
- Digitální podpisy
- Záložní kopie dat
- Mechanismy řízení přístupu
- Alternativní zdroje energie atd.

2.1.8 Zbytkové riziko

Riziko není často ochranným opatřením plně eliminováno, ale většinou se jedná o částečné zmírnění rizika. Je zřejmé, že čím větší zmírnění chceme dosáhnout, tím více nákladů musíme vynaložit. Existuje například mnoho úrovní bezpečnosti informačního systému. A pokud mají aktiva (např. data v databázi) menší finanční hodnotu, než zavedení vysoké úrovně bezpečnosti, tak zavedeme takovou úroveň, která nám dostačuje a především nepřesahuje cenu aktiv. Z toho vyplývá, že často existují zbytková rizika, která organizace posuzuje a pokud odpovídají potřebám organizace, tak jsou přijímány. Tento proces se nazývá akceptace rizik.

Vzhledem k tomu, že existuji zbytková rizika, je důležité, aby o nich byl informován management a rozhodoval o akceptaci těchto rizik vždy ten, kdo je oprávněn akceptovat dopad nežádoucích incidentů. Dělá se to především proto, aby byl daný manažer na tyto zbytkové rizika lépe připravený, nebo v případě že nebude chtít zbytkové riziko akceptovat, mohl implementovat nové ochranná opatření.

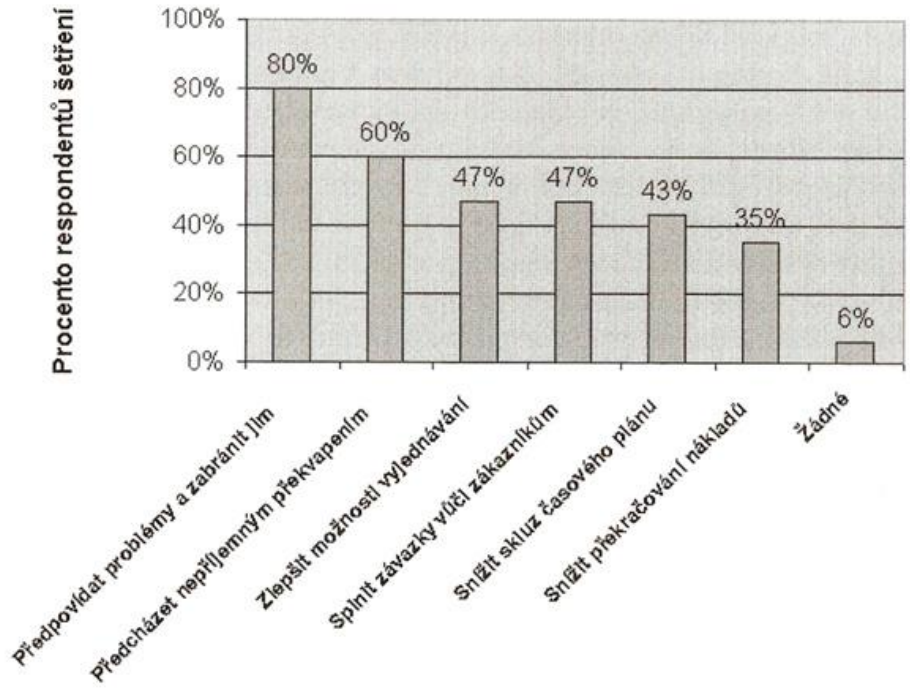
2.2 Úvod do řízení rizik v projektu

S ohledem na unikátnost každého projektu jsou s projektem a jeho realizací spojena rizika, jejichž řízení je jednou z klíčových částí řízení projektu. Na řízení rizik v projektech existuje spoustu postupů a metodik, ale pro správné a efektivní řízení rizik musí mít projektový manažer velké zkušenosti z praxe spolu s dobrou předtuchou, a proto se řízení rizik z částí považuje také za umění. Řízení rizik zahrnuje identifikaci rizik, jejich analýzu, reakci na ně a následný monitoring rizik. Tyto části řízení by se měly provádět po celou dobu života projektu a v případě nutnosti aktualizovat a průběžně upřesňovat dříve identifikovaná rizika.

Řízení rizik bývá v projektech často přehlíženo nebo na něho nezbývají zdroje jak finanční tak lidské. Málo kdo si ale uvědomí, že při správném řízení rizik můžou přijít pozitivní dopady jak při výběru projektu, tak stanovení jeho rozsah, stanovení realistického časového plánu a odhadu nákladů. Díky řízení rizik, se lépe definují slabé a silné stránky projektu, do kterých jsou zapojeni i ostatní členové týmu. Tím můžeme přispět k lepšímu poznání projektu pro celý tým.

Dobré řízení rizik bývá někdy těžké na rozdíl od krizového řízení ohodnotit. Při krizovém řízení vniká přímé nebezpečí pro dokončení projektu. Proto tak na sebe upoutá větší pozornost celého týmu čí organizace. Je zřejmé, že vyřešení krize je pak více viditelné, než úspěšné řízení rizik. Ale je potřeba si uvědomit, že efektivní řízení rizik snižuje počet problémů. Menší počet problémů jsme schopni rychleji vyřešit a předcházíme tak krizím.

Výzkumná společnost KLCI Research Group provedla v roce 2001 šetření mezi 260 organizacemi zaměřenými na vývoj softwaru. Již tehdy vidělo 80% organizací jako hlavní přínos řízení rizik především předpokládání a zabránění problémů. Kromě předpovídání a zabránění problémů viděli organizace sílu v předcházení nepříjemných překvapení, zlepšení vyjednávání, splnění závazků vůči zákazníkovi, snížení skluzu časového plánu a překračování nákladů. Obrázek 2.1 shrnuje hlavní přínosy řízení rizik ze zmíněného výzkumu [4].



Obrázek 2.1: Výhody zavedení praktických postupů řízení rizik při vývoji softwaru [4]

Jak již bylo nastíněno v kapitole 2.1.6, riziko nemusí být vždy negativní. Existují také pozitivní rizika, která v projektu vedou k příznivým důsledkům. Z toho vyplívá, že riziko může mít záporný ale i kladný vliv na splnění projektu. Řízení negativního rizika pak můžeme přirovnat k práci s pojištěním projektu a pozitivní riziko lze srovnávat s investiční příležitostí. Na cíl řízení rizik v projektu můžeme pohlížet jako na minimalizaci negativních rizik a maximalizaci pozitivních rizik.

Odborníci doporučují, aby se organizace pokusily najít jakousi rovnováhu mezi riziky a příležitostmi, což tak ve většině případů také organizace dělají. Lidé vnímají tuto rovnováhu ale různě. Z toho vyplývá, že organizace mají k rizikům rozdílný přístup. Některé organizace mají k rizikům neutrální postoj, jiné rizika vyhledávají, anebo k nim mají averzi. Z těchto tří typů preferencí rizika se odvozuje teorie o užitku rizika neboli tolerance k riziku.

Množství uspokojení nebo výhod, které můžou vzniknout z tolerance k riziku je znázorněno na obrázku 2.2. Jsou zde vidět rozdíly mezi lidmi resp. organizacemi, kteří jsou k rizikům neutrální, nebo mají k rizikům averzi, nebo naopak rizika vyhledávají. Graf funkce znázorňuje užitek neboli míru uspokojení, plynoucí z podstoupení rizika vůči objemu potenciálních přínosů (např. peněžní hodnota rizikové příležitosti). Graf znázorňující averzi k riziku, tedy nízkou toleranci k riziku ukazuje, že růst užitku se s rostoucím množství přínosů snižuje. Lze tedy říci, že pokud má organizace, která má averzi k riziku možnost získat z rizika více přínosů (peněz), získá tak méně uspokojení a užitku. Naopak, pokud organizace rizika vyhledává, tak se při rostoucích přínosech zvyšuje i míra užitku. To znamená, že tento typ organizací sáhne například po projektu, který je vysoce rizikový, ale při dokončení takového projektu získá organizace vysoký výnos a velký užitek. Rovnováhu mezi těmito dvěma variantami tvoří organizace rizikově neutrální. Ta se snaží najít rovnováhu mezi přínosy a zisky.

Obrázek 2.2: Funkce užitku z rizika při různých preferencích rizik [4]

Je zřejmé, že ne všechny rizika může projektový tým řídit. Existují rizika, která již byla dříve nalezena resp. identifikována a analyzována a ty nazývají „známá rizika“. Existují ale také rizika, která neznáme, tedy ještě jsme je neidentifikovali, ani neanalyzovali. Z toho vyplývá, že je velmi důležité, snažit se předem identifikovat všechny potenciální rizika a řídit je. Řízení rizik se skládá z celkem šesti hlavních procesů, mezi které patří:

- Plánování řízení rizik
- Identifikace rizik
- Kvalitativní analýza rizik
- Kvantitativní analýza rizik
- Strategie zvládnutí rizik
- Monitoring a kontrola rizik

Jednotlivé části řízení rizik jsou rozepsány v následujících kapitolách.

2.3 Plánování řízení rizik

Plánování řízení rizik je proces, ve kterém se rozhodujeme, jak budeme postupovat při aktivitách řízení rizik v projektu a jak je budeme plánovat. Jako výstup tohoto procesu je plán řízení rizik, který popisuje postupy řízení rizik v projektu. Tento plán by měl vzniknout za pomocí projektového týmu hned na začátku životního cyklu projektu. Hlavní náplní plánování je zrevidování projektových dokumentů, podnikových zásad, rozdělení rizik do kategorií, prozkoumání zpráv s poučením z předchozích projektů a šablon pro vytvoření plánu řízení rizik. Je v hodné se zaměřit také na toleranci k rizikům u zadavatele a ostatních členů projektu. Pokud má zadavatel averzi k rizikům, budeme muset zřejmě nastavit jiný přístup k řízení rizik v projektu, než kdyby zadavatel rizika vyhledával.

Z pohledu plánováni projektu je plán řízení rizik jen částí celkového plánu řízení projektu. Podle potřeb konkrétního projektu budeme také vykonávat úroveň detailu plánu řízení rizik. V tabulce 2.1 je shrnuto několik obecných témat, kterými by se měl plán řízení rizik zabývat. Jedná se především o ujasnění si rolí jednotlivých členů projektu, rozdělení si povinností, připravení odhadu rozpočtu a časového plánu na práce související s riziky a identifikovat kategorie rizik. Dále musíme

určit, jak budeme hodnotit pravděpodobnosti a dopady rizik, a jaký formát budou mít dokumenty související s riziky [4].

Metodologie:	Jak budeme v projektu provádět řízení rizik? Jaké nástroje a zdroje dat máme k dispozici a jaké použijeme?
Role a povinnosti:	Kteří konkrétní jednotlivci jsou odpovědní za implementaci konkrétního úkolu a za vytvoření konkrétních ucelených částí projektu souvisejících s řízením rizik?
Rozpočet a časový plán:	Jaké odhadované náklady a časový plán budou vyhrazeny na provádění aktivit souvisejících s riziky?
Kategorie rizik:	Jakými hlavními kategoriemi rizik se budeme v tomto projektu zabývat? Existuje v projektu struktura rozdělení rizik?
Pravděpodobnost a důsledky rizik:	Jak budeme hodnotit pravděpodobnost a dopady rizikových položek? Jaké metody použijeme při tvorbě kvalitativní a kvantitativní analýze rizik?
Dokumentace rizik:	Jaké formáty zpráv a jaké procesy použijeme při aktivitách řízení rizik?

Tabulka 2.1: Obecná témata v plánu řízení rizik [4]

K plánu řízení rizik se někdy také vytváří mimořádný plán a havarijní plán. Mimořádný plán popisuje, jak budeme postupovat v případě, že nastane nějaká riziková událost. Havarijní plán nám zase říká, jak budeme postupovat, pokud aplikujeme definované mimořádné plány a ty selžou.

Pro správné pochopení zbylých částí procesů v řízení rizik je vhodné nejprve poznat nejběžnější zdroje rizik, o kterých je psáno v další kapitole.

2.4 Nejčastější zdroje rizik v projektech z IT

Je samozřejmostí, že projekty zabývající se informačními technologiemi, mají vždy několik společných rizik. Proto je možné stanovit si několik hlavních kritérií úspěchu a ke každému z kritérií doplnit upřesňující otázky. Mezi hlavní kritéria patří například zapojení uživatelů, podpora firemního vedení, jasně stanovené požadavky, odborně zdatní zaměstnanci, jasné cíle a vize, atd. Kritérium zapojení uživatelů může souviset s následujícími otázky:

- Máme ty správné uživatele?
- Máme s uživateli dostatečné vztahy?
- Je nám jasné co uživatel požaduje?
- Je uživatel dostatečně zapojován do řešení projektu?

Podle získaného hodnocení kladných a záporných odpovědí se můžeme lépe rozhodnout před započetím projektu. Například zda je možné daný projekt dokončit, anebo ho raději vůbec neřešit, popřípadě nejprve přijmout vhodná opatření pro snížení rizik a teprve poté, do projektu začít investovat náklady.

Organizace si většinou vytvářejí vlastní dotazníky rizik, ve kterých se zaměřují na tyto kategorie rizik:

- Tržní rizika
- Finanční rizika
- Technologická rizika
- Lidská rizika
- Strukturální a procesní rizika

V případě, že výsledkem projektu bude nový produkt nebo služba, můžeme si v tržních rizicích například pokládat otázky, jestli uživatelé nový produkt přijmou a zvyknou si na něj, zda nemůže někdo jiný vytvořit podobný produkt dříve a lepší, nebo zda bude nový produkt přínosem pro organizaci.

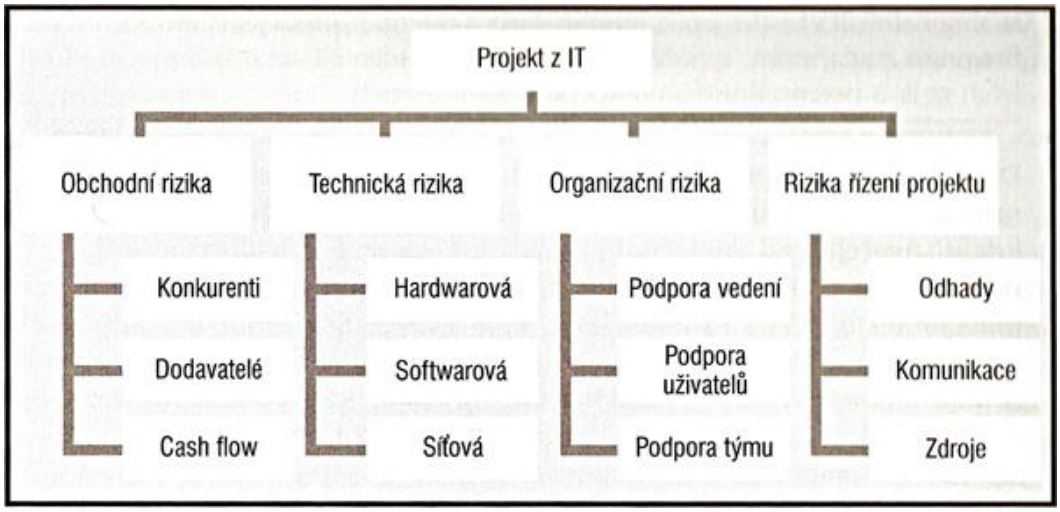
Ve finančních rizicích se zase řeší otázky, jestli si může organizace dovolit financování projektu, jestli je možné důvěřovat finančním odhadům, zda je zajištěna návratnost investic atd.

Technologická rizika pojednávají o tom, zda je projekt technicky zvládnutelný, jaké typy technologií se budou využívat, zda zastaralé nebo moderní špičkové, nebo nové ale nevyzkoušené. Zda se nestanou technologie zastaralé ještě před dokončením projektu, nebo naopak, zda bude potřebná technologie včas k dispozici atd.

V kategorii lidských rizik si můžeme položit otázky typu, jestli má organizace dostatek pracovníků s odpovídajícími schopnostmi. Zda tito lidé mají dostatek zkušeností v praxi. Jestli dostatečně známe zákazníka projektu, a jaké vztahy s tímto zákazníkem máme.

Strukturální a procesní rizika se zabývají například otázkami, kolik různých skupin uživatelů musí tento projekt uspokojit? Jestli bude náš produkt spolupracovat s dalšími systémy a kolik jich bude? Má organizace zavedené příslušné procesy pro dokončení projektu?

Dalším užitečným nástrojem pro projektového manažera je struktura rozdělení rizik. Tato technika spočívá v rozdělení potenciálních rizik do různých kategorií. Výsledná podoba této hierarchie má podobnou strukturu jako rozdělení nebo rozpis prací. Ukázka toho jak by mohlo vypadat rozdělení rizik v projektech z informačních technologií je na obrázku 2.3.



Obrázek 2.3: Příklad kategorizace rizik v projektech z informačních technologií [4]

Podle literatury [4] by kategorie rizik měly být rozděleny na obchodní, technická, organizační a rizika řízení projektu. Tyto kategorie se pak můžou dále rozdělit. Například pod obchodní rizika patří konkurenti, dodavatelé. Pod technické záležitosti patří zase rizika hardwarová, softwarová a síťová atd.

Pokud správně porozumíme nejběžnějším zdrojům rizik, jsme schopni identifikovat rizika. Následující kapitola popisuje, jak tato identifikace probíhá.

2.5 Identifikace rizik

Identifikace rizik je proces, ve kterém určujeme rizika, která mohou ovlivnit projekt a to jak kladně tak záporně. Rizika a jejich charakteristiky se zapisují do dokumentace a jsou v průběhu projektu doplňovány o nová vzniklá rizika podle potřeb měnícího se prostředí. Zdrojem informací pro identifikaci rizik může být smlouva a další související dokumenty, firemní plány, kontrolní zprávy, výsledky minulých projektů podobného typu atd. Nejdůležitějším prvkem v procesu identifikace rizik je začít s rozpoznáváním rizik včas. Provádět identifikaci pravidelně a často v ní pokračovat a to na všech úrovních.

2.5.1 Techniky pro identifikaci rizik

Pro identifikaci rizik existuje několik postupů. V počátku se většinou jedná o diskuzi mezi členy projektového týmu s pozvanými externími odborníky na danou problematiku. Jakmile se v této diskuzi vyčlení základní okruh potencionálních rizik, použijí se další techniky pro nalezení nových rizik. Mezi tyto techniky patří brainstorming, delfská metoda, rozhovory, analýzy prvotních příčin a SWOT analýza.

2.5.1.1 Brainstorming

Brainstorming je technika zaměřená na generování co nejvíce nápadu. Tato technika se zásadně používá ve skupině několika lidí. Myšlenka této metody je předpoklad, že skupina lidí dokáže vymyslet více na základě podmětu ostatních, než by vymysleli samostatně. Když bude brainstorming správně použit, může pomoci vytvořit dlouhý seznam potenciálních rizik. U této techniky by měl být vždy moderátor, který povede diskuzi a bude oživovat myšlenky. Dále jsou potřeba zapisovatelé všech myšlenek. Žádné myšlenky by neměly být na místě nikým komentovány ani hodnoceny, protože někdy může mít v sobě i zdánlivě hloupý nápad inspirativní myšlenku. Podle některých výzkumů je brainstorming kritizován. Bývá poukazováno na to, že u brainstormingu může být vygenerováno méně nápadů, než kdyby členové pracovali samostatně. To může být zapříčiněno strachem některých účastněných, nebo blokováním výkonu tím, že mluví vždy pouze jeden a druzí mlčí, čímž dochází k zapomínání nápadů ostatních. Další nevýhodou může být také to, že členové nejsou hodnoceni za počet nápadů, protože jsou nápady sepisovány dohromady a to by mohlo mít vliv na sociální lenivost [5].

2.5.1.2 Delfská metoda

Delfská metoda je postup, jenž pomáhá překonávat některé negativní efekty z brainstormingu. Cílem této metody je dosažení shody mezi všemi členy skupiny. Metoda se provádí v několika opakovaných kolech za přítomnosti expertů, kteří písemně odpovídají na otázky, mezi které můžou patřit i připomínky k odpovědím z předchozích kol. Tyto odpovědi pak sumarizuje prostředník a

distribuuje je do dalšího kola. Tohle se opakuje tak dlouho, dokud skupina nedospěje k jednotnému konkrétnímu řešení.

2.5.1.3 Rozhovor

Rozhovor je běžná technika, která slouží ke shromaždování informací. Probíhat může osobním setkáním nebo elektronicky. Jedná se o dotazování lidí, kteří mají zkušenosti z podobného problému. Například pokud máme pracovat s nějakým novým hardwarem, můžeme se zeptat někoho, kdo již s tím to hardwarem pracoval a získat tak seznam potenciálních rizik. Podobně můžeme postupovat u získávání informací o zákazníkovi, s kterým jsme ještě neobchodovali.

2.5.1.4 SWOT analýza

SWOT (**S**trengths, **W**eaknesses, **O**pportunities, **T**hreats) je analýza silných a slabých stránek, příležitostí a hrozeb dané firmy, která se používá převážně v marketingu, ale lze ji mimo jiné také použít na identifikaci rizik v projektu. Díky této metodě identifikujeme hrozby, což může být například konkurence, která by měla větší šanci při získání projektu, nebo naopak lze zjistit, že díky tomuto projektu si firma otevře dveře k dalším zakázkám.

2.5.2 Registr rizik

Výstupy z procesu identifikace rizik, jež jsou seznam rizik a jejich charakteristika, se použijí k tvorbě dokumentu registru rizik. Tento dokument bývá často znázorněn ve formě tabulky, která obsahuje potenciální rizikové události. V tabulce 2.2 je znázorněna ukázka registru rizik.

Číslo	Stupnice	Riziko	Popis	Kategorie	Prvotní příčina	Spouštěče	Potenciální reakce	Vlastník rizika	Pravděpo-dobnost	Dopady	Stav
R44	1										
R21	2										
R7	3										
...											

Tabulka 2.2: Ukázka registru rizik [4]

Jak je vidět, charakteristika rizika v registru rizik je podstatně obsáhlá, což vypovídá o tom, že tento proces bude časově náročný. Pro správný a pravděpodobnostně pravdivý registr rizik, budeme při jeho tvorbě potřebovat řadu expertů pro dané specifické riziko. Je důležité upozornit, že v této fázi se pravděpodobnost rizika zatím určuje slovně (např. častá, malá, střední) nebo přiřazenou číselnou hodnotou, která je vypočtena v kvalitativní analýze, o které je zmínka v další kapitole. Upřesnění pravděpodobnosti se bude provádět až v kvantitativní analýze, o které bude zmíněno později.

2.6 Kvalitativní analýza rizik

Kvalitativní analýza se vyznačuje tím, že míra rizika neboli jejich dopady jsou vyjádřeny v určitém rozsahu hodnot (např. <1 až 10>, nebo pravděpodobností <0,1> nebo slovně). Charakteristika dopadu je tedy znázorněna pouze kvalitativně. Tato metoda je sice rychlejší, ale také více subjektivní. Pomocí této metody neposoudíme například přesné finanční náklady nutné pro eliminaci hrozby, protože bychom touto metodou získali např. charakteristiku ve tvaru „velká až

kritická“. Pokud bychom chtěli přesnější analýzu, museli bychom použít některou z kvantitativních metod. Hlavním výstupem kvalitativní analýzy rizik je aktualizovaný registr rizik doplněný o hodnocení rizika.

2.6.1 Nejčastější techniky používané v kvalitativní analýze

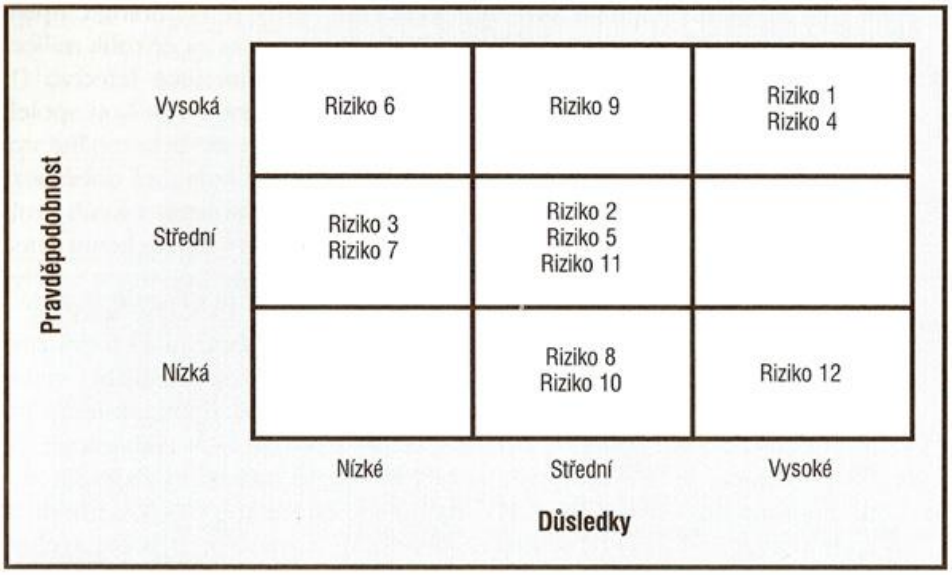
Metody v kvalitativní analýza nám většinou řeknou slovně jaké riziko má jakou pravděpodobnost a jaký dopad vůči této pravděpodobnosti. Běžné metody, které se používají, jsou:

- matice neboli diagram pravděpodobnosti a důsledků
- sledování deseti nejrizikovějších položek [4]

O ostatních metodách využívané v kvalitativní ale také v kvantitativní analýze bude zmíněno ještě v kapitole 3.

2.6.1.1 Matice pravděpodobnosti a důsledků

Tato technika je znázorněna ve formě diagramu, který na jedné straně uvádí relativní pravděpodobnost vzniku rizika a na druhé relativní důsledky na vznik rizika. Díky této metodě okamžitě vidíme, na která rizika si musíme při tvorbě projektu dávat větší pozor. Technika je v celku jednoduchá. Po uvedení všech rizik v projektu, přiřadíme k rizikům pravděpodobnost vzniku (vysoké, střední, nízké) a velikost důsledku (vysoké, střední, nízké). Tyto zjištěné výsledky zapíšeme do matice pravděpodobnosti a důsledků, která může vypadat podobně jako na obrázku 2.4.



Obrázek 2.4: Příklad matice pravděpodobnosti a důsledků [4]

V některých případech může být užitečné vytvořit matici pravděpodobnosti a důsledků pro negativní a pro pozitivní rizika. Pomocí takto vytvořených dvou digramů se lze lépe orientovat mezi riziky a nezapomeneme na ně.

2.6.1.2 Sledování deseti nejrizikovějších položek

Tato technika navíc od identifikace rizik udržuje podvědomí o nejvíce rizikových událostech, po celou dobu projektu. Jedná se o tabulku rizik, které jsou podrobeny pravidelným revizím. Tabulka odpovídá seznamu nejzávažnějších rizik v daném projektu. Vedle názvu rizika je v tabulce uvedeno aktuální hodnocení rizika, předchozí hodnocení rizika, kolikrát se již riziko objevilo na seznamu deseti nejrizikovějších položek a jakého bylo dosaženo postupu při řešení tohoto rizika od poslední revize. Díky těmto revizím jsou vedení firmy a také zákazník (pokud je zapojen do jednání) průběžně informováni o nejdůležitějších vlivech, které mají potenciál podpořit, nebo ohrozit dokončení projektu. Je tak posílena důvěru v projektovém týmu a u zákazníka, protože ukážeme, že jsme si vědomi toho, co může projekt ovlivnit, a že aplikujeme strategii, která rizika minimalizují, nebo naopak (pokud se jedná o kladná rizika) maximalizují.

2.7 Kvantitativní analýza rizik

Kvantitativní metody analýzy jsou založeny na matematickém výpočtu z frekvence respektive pravděpodobnosti výskytu hrozby a jejího dopadu. Většinou se dělají jako rozšíření pro kvalitativní analýzu, i když oba procesy můžou probíhat současně. Kvantitativní metody jsou více exaktní než kvalitativní, což se odráží na složitosti metod a vysokým časovým nákladem. Hlavní náplní kvantitativní analýzy je shromaždování dat (informací) a vlastní kvantitativní analýza a modelování. Výsledkem analýzy pak může být dopad rizika na rozpočet projektu, vyjádřený přímo v hodnotách měny, jako například tisíce Kč, nebo pravděpodobnost dosažení určitých rámcových cílů projektu, atd. V některých případech může být projekt na základě kvantitativní analýzy pozměněn, nebo úplně zastaven. Jindy jsou zase zahájeny další projekty, které mají dopomoci aktuálnímu projektu pro úspěšné dokončení.

Techniky používané v tomto procesu řízení rizik jsou poměrně složité, a proto jsou popsány v samostatné kapitole 3.

2.8 Plánování reakcí na rizika

Plánování reakcí na rizika neboli strategie zvládnutí rizik se týká definování odezev na hrozby, nebo definování opatření na využití příležitosti. Jedná se o vypracování možných variant reakcí pro jednotlivá identifikovaná rizika. Hlavním výstupem v této části procesu jsou smluvní ujednání související s riziky, aktualizovaná verze plánu řízení projektu a registru rizik. Při tvorbě reakcí na rizika se zaměřujeme na to, abychom posílili pozitivní rizika a oslabili negativní rizika. Na oslabení negativních rizik můžeme použít následující strategie [4]:

Zabránění riziku – jedná se o potlačení hrozby nebo rizika tím, že potlačíme jeho příčiny. Můžeme toho dosáhnout například tak, že nebudeme využívat nový software, u kterého nemáme ještě ověřenou správnost výpočetních funkcí, ale použijeme starší ověřenou verzi softwaru i na úkor časových nároků.

Přijetí rizika – riziko si můžeme dovolit přijmout v případě, že nenese velké ztráty, anebo není jiná možnost.

Přenos rizika – spočívá v přenesení rizika na třetí osobu/subjekt. Může se jednat například o pojištění hardwaru. Tedy, že ho výrobce musí do určité doby vyměnit nebo zaplatí případné ztráty.

Potlačení rizika – pro případ snížení pravděpodobnosti vzniku rizika, můžeme riziko potlačit. Potlačit vznik rizika jde například tím, že budeme dělat pravidelnou údržbu hardwaru, zlepšíme komunikaci v projektu nebo zvýšíme autoritu manažera projektu.

Na pozitivní rizika pak jde reagovat pomocí následujících strategií:

Využití rizika – riziko můžeme vyžít například v posílení dobrého mínění o firmě tím, že po dobře ukončeném projektu necháme napsat článek do novin nebo uspořádat veřejnou událost.

Sdílení rizik – sdílení spočívá v přidělení vlastnictví rizika jiné osobě/subjektu. Může se jednat například o spojení se s druhou firmou, která nám pomůže proslavit náš projekt.

Posílení rizika – riziko můžeme posílit tím, když se nám podaří nadchnout pro myšlenku projektu více lidí, kteří následně udělají formální i neformální reklamu, která může organizaci přinést další projekty od nových zákazníků.

Přijetí rizika – jedná se o prosté přijetí pozitivního rizika. Pokud například uděláme výjimečný projekt je zřejmé, že ho budou uživatelé chválit i bez toho, abychom do něj nějak dále zasahovali.

2.9 Monitoring a kontrola rizik

Monitorování neboli sledování a kontrola rizik je proces, který reaguje na konkrétní rizikové události. Pokud dojde ke změnám plánu protirizikových opatření, nebo se objeví nové riziko, musí se opakovat základní cyklus, který spočívá ve stanovení, ohodnocení a reagování na rizika. Monitorování rizik by měl zabezpečit, že projektový tým si bude udržovat v podvědomí rizika po celou dobu řešení projektu. Musí se tato činnost ale stát pravidelnou aktivitou v projektovém týmu. V průběhu sledování rizik můžeme zjistit, že některá rizika se vůbec neprojeví, nebo se pravděpodobnosti jejich vzniku změní, popřípadě u některých rizik může hrozit vyšší odhadované ztráty. Tyto poznatky se nám pak můžou hodit při řešení podobného projektu.

Jako nástroje pro sledování a kontrolu rizik můžeme použít audity rizik, analýzu odchylek a trendů, analýzu rezerv. Mezi hlavní činnosti patří pravidelné porady s hodnocením a revizemi stavu rizik. Hlavním výstupem v tomto procesu jsou požadované změny, preventivní opatření, aktualizované verze registru rizik, dokumentace s informacemi, které můžeme použít v řešení nových projektů.

3 Techniky používané v analýze rizik

Tato kapitola se zabývá popisem několika významných metod pro kvalitativní a především pro kvantitativní analýzu rizik. Zaměřuje se na popis analýzy stromu chyb a stromu událostí. Pro analýzu a hodnocení rizik je v současnosti k dispozici řada metodik, ke kterým je vyvíjeno velké množství softwarových nástrojů. Tyto metody byly vynalézány postupně k určitým projektům a problémům, proto se ne každá metoda hodí na libovolný projekt. Projektový manažer nejprve musí podle žádoucího cíle hodnocení rizik zjistit, zda jsou splněny předpoklady, požadavky a datové úložiště pro danou metodu. Nicméně je nutné konstatovat, že žádná metoda nedokáže přesně předpovědět vývoj a pravděpodobnost rizik v projektu. Tyto metody pouze pomáhají analyzovat rizika ve složitějších projektech například pomocí softwarového řešení dané metody. Vždy tuto aplikaci ale obsluhují lidé, kteří nejsou bezchybní.

3.1 Analýzu stromu chyb (FTA)

3.1.1 Všeobecné informace o FTA

FTA (Fault Tree Analysis – analýza stromu chyb) je deduktivní metoda a zabývá se identifikací a analýzou podmínek, které způsobují, nebo mohou způsobit výskyt vrcholové události. Vrcholová událost bývá porucha, vada zhoršení bezpečnosti atd. Tato metoda je vhodná pro komplikovanější systémy, kde dochází k větvení a paralelním procesům. Metoda se začala používat při analýze rizik v jaderných elektrárnách, chemických průmyslech, organizacích zabývajícími se komunikačními systéme a postupně se rozšířila i do dalších odvětví. Tato metoda byla vyvinuta hlavně pro analýzu bezpečnosti nebo pohotovosti a udržovatelnosti systému. FTA se využívá jak pro kvalitativní, tak pro kvantitativní analýzu. Kvalitativní metoda se používá tam, kde se hledají pouze potenciální příčiny poruchových stavů neboli rizik. Naopak kvantitativní metoda se používá k modelování celého produktu, procesu nebo systému. Vstupní události mají zadanou pravděpodobnost/frekvenci poruchy, se kterými lze posléze spočítat pravděpodobnost vrcholové události.

3.1.2 Termíny a definice

Základní používané termíny k metodice FTA jsou v různých odborných publikacích odlišné. Já jsem převzal tyto termíny z normy ČSN EN 61025 zabývající se analýzou stromu poruchových stavů [6]. Zde je soupis těch nejdůležitějších:

výstup – výstup děje či jiného vstupu

událost – výskyt určité podmínky nebo děje

vrcholová událost – výstup kombinací všech vstupních událostí
Poznámka: Jedná se o událost, která je předmětem zkoumání, pod kterou se rozvíjí strom poruchových stavů.
Poznámka 2: Někdy tato událost bývá nazvaná jako konečná událost nebo vrcholový výstup

primární událost – událost, která se nachází na základní úrovni FTA

základní událost – událost, která se dále nerozvíjí

mezilehlá událost – událost mezi vrcholovou a primární

nerozvíjená událost – událost bez vstupních událostí

události se společnou příčinou – odlišné události, které mají stejnou příčinu svého výskytu.
Poznámka: Příkladem takové události může být zkrat kondenzátorů v důsledku prohnutí desky plošného spoje. Kondenzátory můžou mít jinou funkci, ale zkrat by měl stejnou příčinu (vstupní událost) pro oba kondenzátory.

několikanásobná nebo opakovaná událost – událost, která je vstupem pro více než jednu událost na vyšší úrovni FTA

hradlo – značka, která se používá ke znázornění vazby mezi výstupní událostí a odpovídajícími vstupy

kritický řez – skupina událostí, které při současném výskytu způsobí výskyt vrcholové události

3.1.3 Rozvoj a vyhodnocení stromu poruchových stavů

Pokud chceme analyzovat nějaký systém pomocí FTA měly bychom si definovat nebo obstarat následující body:

- Souhrnný přehled záměru systému
- Vymezit, co je podstatou poruchy systému
- Funkční strukturu systému pro lepší porozumění analyzovanému systému (většinou bývá znázorněna pomocí nějakého diagramu)
- Hranice systému, jako jsou elektrická, mechanická a provozní rozhraní, které jsou ovládány vzájemným působením s jinými systémy.
- Fyzickou strukturu systému
- Provozní režim, popis a fungování systému v každém provozním režimu
- Provozní profil systému
- Lidská hlediska (např. stupeň výcviku pracovníků)
- Seznam použitelných dokumentů (např. výkresy, provozní příručky) [6].

3.1.4 Sestavení stromu poruchových stavů

Strom poruchových stavů bývá reprezentován graficky ve formě rozvinutého stromu a může být zakreslen buď svisle, nebo vodorovně. Pokud je zakreslen svisle měla by být vrcholová událost nahoře na stránce a základní události dole. Pokud je strom zakreslen vodorovně může být vrcholová událost vlevo nebo vpravo na stránce.

Základní rozdíl FTA od jiných metod je ten, že do FTA se zahrnují pouze ty události, které přispívají k výskytu vrcholové události. Mezi událostmi modelujeme funkční kombinaci a

dynamickou interakci a závislosti. Ostatní metody se zabývají intenzitami nebo pravděpodobností poruch součástí s předpokladem nezávislosti poruch. U FTA může být analyzovaný produkt složen s několika na sobě závislých součástek (které mají jinými metodami ohodnocenou pravděpodobnost poruchy). Takovýto produkt může být například porušen právě tehdy, pokud je porušena jen určitá jeho jedna součástka (komponenta) nebo několik jiných součástí. Pravděpodobnost poruchy celého produktu by tedy byla podle pravděpodobnosti dané součástky nebo součástí. Tato technika tedy umožňuje návrhářovi produktu realistickou identifikaci těch způsobů poruch, na které je potřeba dávat větší pozornost.

3.1.4.1 Sériová konfigurace systému

Při modelování používáme sériovou konfiguraci právě tehdy, když by porucha libovolné součástky z této sestavy znamenala poruchu celého systému. Model ve stromu poruchových stavů je takový, že všechny události a hradla vstupují do hradla *OR* (viz obrázek 3.1).
Matematický výraz pro pravděpodobnost bezporuchového provozu vrcholové události s takovouto konfigurací skládající se z „n“ nezávislých bloků je následující (níže uvedené vzorce jsou převzaty z [6]):

$$R_s(t) = R_1(t) \cdot R_2(t) \cdot \dots R_i(t) \cdot \dots R_n(t) \quad (1)$$

Výše uvedený výraz v podobě pravděpodobnosti nám říká, že blok 1 i blok 2 i ostatní bloky musejí být v provozu, aby byl celý systém také v provozuschopném stavu.

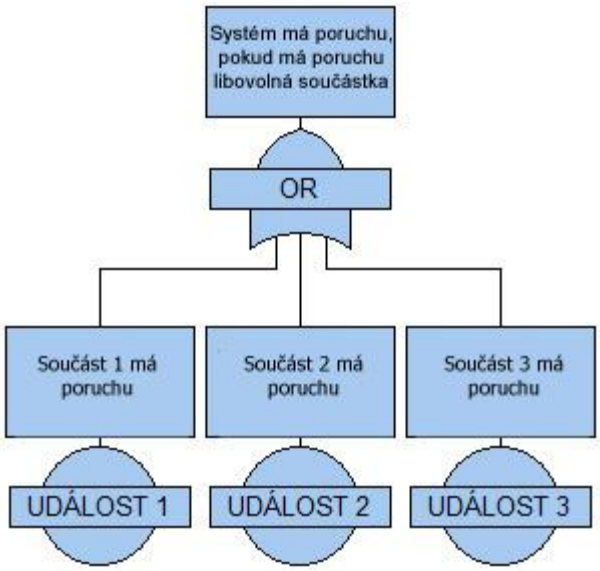
V FTA se často používá místo pravděpodobnosti bezporuchovosti systému, také pravděpodobnost poruchy $F(t)$, což je pravděpodobnostní doplněk k pravděpodobnosti bezporuchového provozu.

$$F(t) = 1 - R(t) \quad (2)$$

Pravděpodobnost poruchy systému reprezentovaného hradlem *OR* skládajícího se z „n“ nezávislých vstupních hradel nebo událostí je:

$$F_s(t) = 1 - [1 - F_1(t)] \cdot [1 - F_2(t)] \cdot \dots [1 - F_i(t)] \cdot \dots [1 - F_n(t)] \quad (3)$$

Systém má poruchu, když má jakákoliv součástka poruchu.



Obrázek 3.1: Ukázka sériové konfigurace systému pomoci FTA [6]

3.1.4.2 Paralelní konfigurace systému – aktivní záloha

Paralelní konfigurace se používá tehdy, když výstupní událost hradla nastane pouze tehdy, nastanou-li všechny vstupní události. Pro toto spojení událostí se používá hradlo *AND*. Pro zjednodušení budu nyní uvažovat, že každý vstup do bloku zůstává stejný, nezávisle na tom, kolik jiných vstupů pracuje a také se předpokládá, že každý vstup je nezávislý. Nezávislost je splněna tehdy, když se pravděpodobnost výskytu vstupních událostí nemění bez ohledu na stav ostatních vstupů. Pokud není některý vstup nezávislý, používá se pro výpočet a modelování technika „*vytváření disjunkcí"* [6]. V takovémto případě nelze hradlo *AND* použít, ale používá se dynamické hradlo.

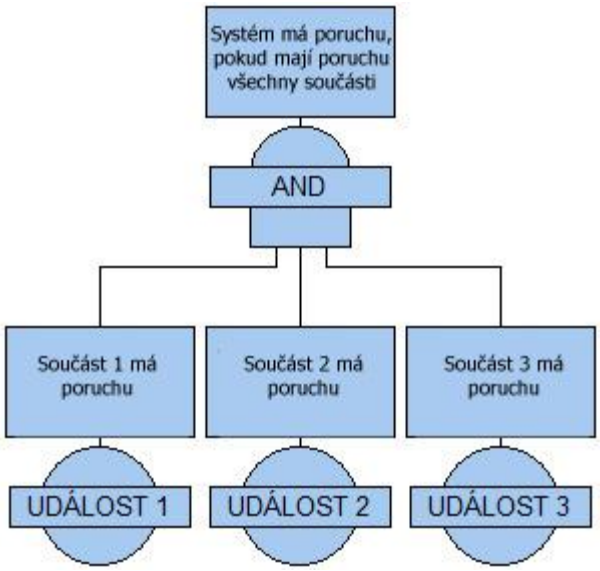
Matematický výraz pro pravděpodobnost bezporuchového provozu, který má „n" vstupních událostí nebo hradel s významem, že systém má poruchu, jestliže součást 1 i součást 2 i všechny ostatní součásti mají poruchu, je:

$$R_s(t) = 1 - \prod_{i=1}^{n}[1 - R_i(t)] \tag{4}$$

Pravděpodobnost poruchy je potom:

$$F_s(t) = \prod_{i=1}^{n}[F_i(t)] \tag{5}$$

Model reprezentující paralelní aktivní zálohy při FTA je uveden na obrázku 3.2.



Obrázek 3.2: Ukázka paralelní konfigurace systému pomoci FTA [6]

V případě, že je zálohování takové, že systém zůstane provozuschopný, pokud zůstane „k“ z „n“ stejných vstupních bloků provozuschopných, používá se pro výpočet pravděpodobnosti bezporuchovosti resp. poruchy následující matematický zápis:

$$R_s(t) = 1 - \sum_{i=0}^{k-1} \frac{n!}{i! \cdot (n-1)!} \cdot [R_0(t)]^i \cdot [1 - R_0(t)]^{n-i} \tag{6}$$

resp.

$$F_s(t) = \sum_{i=0}^{k-1} \frac{n!}{i! \cdot (n-1)!} \cdot [1 - F_0(t)]^i \cdot [F_0(t)]^{n-i} \tag{7}$$

V modelování FTA je tato kombinace událostí reprezentována hradlem typu „*majoritní hradlo*“. Počet událostí m, které musí nastat, aby se událost šířila ve stromě dále, se počítá jako $m = n - k + 1$. Pokud je například požadovaná záloha 2 z 5 událostí, potom je prahová hodnota 3. Nastanou- li tedy 3 vstupní události, znamená to, že by měl systém poruchu.

3.1.4.3 Paralelní konfigurace systému – pasivní záloha

K aktivní záloze je možné do systému zavést ještě zálohu pasivní neboli pohotovostní. Tato záloha se používá v případě, když dojde k poruše několika aktivních záloh, které by zapříčinily poruchu celého systému. V takovém případě se stanou z pasivních záloh, zálohy aktivní, aby převzaly funkci součástí, které mají poruchu. Pasivní zálohy se popisují pomocí speciálního hradla SPARE (aktivace záloh) [6].

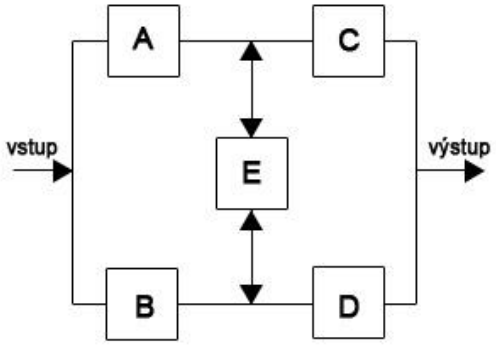
3.1.5 Číselné vyhodnocení stromu poruchových stavů

Cílem číselného vyhodnocení je kvantitativně určit pravděpodobnost výskytu vrcholové události. Proto, abychom mohli provádět číselné vyhodnocení, potřebujeme znát pravděpodobnosti poruch, nebo bezporuchovosti součástí systému označované jako počáteční události.

3.1.5.1 Identifikace minimálních kritických řezů

Kritický řez je skupina událostí, které při současném výskytu způsobí vrcholovou událost. Minimální kritický řez je potom nejmenší taková skupina, ve které musí nastat všechny události, aby nastala vrcholová událost. Určování minimálních kritických řezů, může být u větších stromů obtížné, což dalo podmět k vývoji softwarových aplikací (návrh takové aplikace je součástí této práce), které tuto práci usnadňují.

Pro ilustraci výpočtu FTA jsem použil příklad pro reprezentaci můstkového obvodu, který je znázorněn na obrázku 3.3.



Obrázek 3.3: Příklad můstkového obvodu [6]

Šipky ve výše uvedeném příkladu znázorňují směr signálu. Blokem E může směr signálu směřovat oběma směry. Příklad můstkového obvodu by šel analyzovat jako model s funkčním blokem E, nebo nefunkčním blokem E. V případě funkčního bloku E by mohl jít signál bloky A nebo B a pak C nebo D. Pokud by blok E nefungoval, musel by signál jít bloky A a C nebo B a D. Tento model by šel matematicky zapsat následovně (výpočet je proveden pomocí techniky „*vytváření disjunkcí*“):

$$R_s = (R_A + R_B - R_A \cdot R_B) \cdot (R_C + R_D - R_C \cdot R_D) \cdot R_E + (R_A \cdot R_C + R_B \cdot R_D - R_A \cdot R_B \cdot R_C \cdot R_D) \cdot (1 - R_E) \quad (8)$$

Jestliže $R_A = 0{,}78;\ R_B = 0{,}30;\ R_C = 0{,}15;\ R_D = 0{,}50;\ R_E = 0{,}40,$
Potom $R_S = 0{,}344\ a\ F_S = 0{,}656.$

Kritické řezy by vznikly v případě poruchy kombinací těchto bloků:

- bloky A a B ($c_1 = F_aF_b = ab$)
- bloky C a D ($c_2 = cd$)
- bloky A, E a D ($c_3 = aed$)
- bloky B, E a C ($c_4 = bed$)

Pravděpodobnosti vzniku kritických řezů pak jsou:

$$\begin{aligned} \Pr(c_1) &= F_A \cdot F_B = (1 - R_A) \cdot (1 - R_B) \\ \Pr(c_2) &= F_C \cdot F_D = (1 - R_C) \cdot (1 - R_D) \\ \Pr(c_3) &= F_A \cdot F_E \cdot F_D = (1 - R_A) \cdot (1 - R_E) \cdot (1 - R_D) \\ \Pr(c_4) &= F_B \cdot F_E \cdot F_C = (1 - R_B) \cdot (1 - R_E) \cdot (1 - R_C) \end{aligned} \quad (9)$$

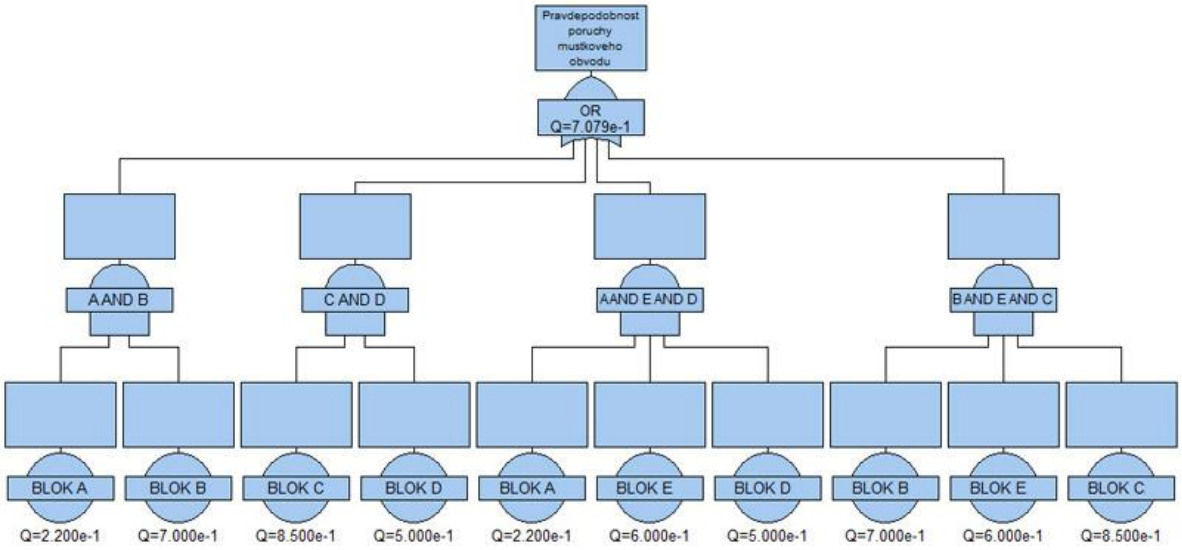
3.1.5.2 Esaryho-Proschanova metoda

Bez vytváření disjunkcí by byla podle Esaryho-Proschanových rovnic pravděpodobnost poruchy příkladu můstkového obvodu (viz obrázek 3.3) následující:

$$F_{S-EP} = \Pr(c_1 \cup c_2 \cup c_3 \cup c_4) \quad (10)$$
$$F_{S-EP} = 0{,}7079; R_{S-EP} = 0{,}2921$$

Výše uvedené výpočty jsou znázorněny pomocí FTA na následujícím obrázku 3.4.

Pokud bychom při výpočtu vrcholové události ignorovali všechny pravděpodobnosti souběžného výskytu kritických řezů, což se často stává v odborných publikacích, provedli bychom výpočet jednoduše součtem pravděpodobností kritických řezů. V takovém případě by ale pro výše uvedený příklad vyšla pravděpodobnost poruchy větší jako 1 (přesný vypočet by byl F = 1,002), což je závažná chyba, protože pravděpodobnost nemůže mít větší hodnotu jako 1 (100%).



Obrázek 3.4: FTA můstkového systému pomocí Esaryho-Proschanových rovnic [6]

3.2 Analýza stromu událostí (ETA)

3.2.1 Všeobecné informace o ETA

ETA (Event Tree Analysis - analýza stromu událostí) je řazena mezi induktivní metody, které provádějí analýzu od příčiny k důsledku. Pomocí této metody identifikujeme možné následky inicializační události. Poruchu začínáme analyzovat od inicializační události a rozvětvujeme ji na další možné následky ve stromu, vždy do dvou větví (příznivý a nepříznivý dopad). Metoda ETA

tedy graficky znázorňuje rozvětvený strom se všemi událostmi (poruchy a chyby) v systému, které se mohou vyskytnout, a které vedou k nehodové události (post-nehodová aplikace), anebo znázorňuje události, které mohou zabránit vzniku nehodových událostí (pre-nehodová aplikace). Frekvence inicializační událost (často únik nebezpečné látky) se může získat pomocí analýzy stromu chyb [2].

Cílem ETA je určit pravděpodobnost události, která je výsledkem k ní chronologicky vedoucích událostí. Z tohoto výsledku pak můžeme určit procento výsledků předem určeného očekávaného dopadu. Výhodou této techniky je univerzálnost aplikovatelnosti. Často tato metoda bývá používána k lepšímu porozumění nákladů životního cyklu projektu ve stavebnictví dále pak v lékařství, ekologii a informačních technologiích [7].

3.2.2 Postup použití ETA

Při provádění analýzy pomocí ETA bychom měli postupovat podle následujících bodů [8]:

- Identifikovat a definovat závažné výchozí události, které mohou mít nepříznivý dopad
- Identifikovat bezpečnostní funkce/nebezpečí podporující faktory
- Vytvořit strom událostí
- Popsat možné výsledky potenciální nepříznivé události
- Definovat frekvenci inicializační události a pravděpodobnost příznivé a nepříznivé větve
- Kvantifikovat výsledky dopadů

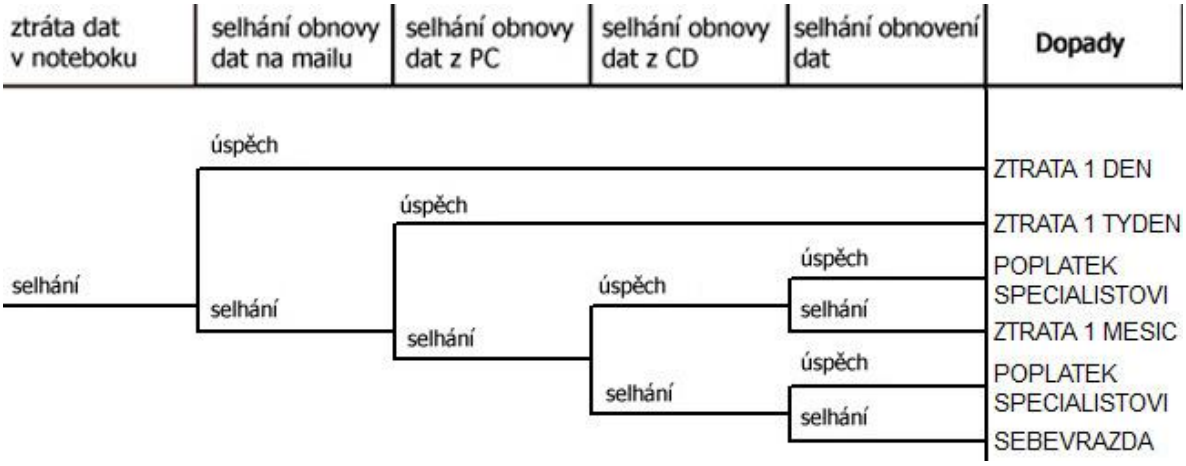
Metoda ETA se používá jak pro kvalitativní, tak kvantitativní analýzu. Pokud bych si určil, že chci analyzovat například mé aktuální riziko, které je ztráta dat v notebooku respektive ztráta tohoto dokumentu, tak podle výše popsaného postupu musím udělat nejdříve kvalitativní analýzu, ve které identifikuji bezpečnostní funkce.

Při psaní tohoto dokumentu můžu mít bezpečnostní funkci např. synchronizaci dat notebooku s osobním PC. Takovouto zálohu jsem schopný provádět jednou týdně. Dále můžu pravidelně zálohovat data na dvě CD s tím, že jedno nechám u PC a druhé budu nosit u notebooku. Tyhle zálohy se snažím vytvářet minimálně jednou do měsíce. Pak provádím zálohy každý večer na spaciální e-mail. Možnou ztrátou by byla, kdyby selhal pevný disk v osobním PC. Je ale ještě možné udělat obnovení dat z disku a tím získat některé data zpět. To znamená, že mám celkem 6 bezpečnostních opatření (záloha dat na osobním PC, dvě DC, e-mail, obnovení disku). Při jaderných reaktorech je standart 5 bezpečnostních opatření. Jak ale ukazuje historie i to někdy nestačí. Například pokud by vypukla mohutná přírodní katastrofa a byly by bezpečnostní prvky blízko u sebe, mohly by se všechny zničit. Hlavním rizikem je ale vždy lidská chyba, protože na lidech jsou závislé všechny ostatní bezpečnostní systémy [9].

Na obrázku 3.5 je zakreslen diagram ETA z výše popsaného přikladu. Strom začíná událostí ztrátou dat z notebooku (např. nechtěným smazáním souboru z disku před uložením). Pokud k tomu dojde, zřejmě si stáhnu zálohu z e-mailu, kterou aktualizuji každý den. Ve stromě událostí se vždy zakresluje uspěná větev směrem nahoru a neuspěná větev směrem dolu. To znamená, že když získám verzi dokumentu z e-mailu, tak směřuju rovnou na konec stromu a dopad je ztráta maximálně jednoho pracovního dne. Pokud selžou data uložená na e-mailu, směřujeme ve stromě událostí směrem dolů, kde narazíme na další bezpečnostní opatření, které je obnovení zálohy

souboru z osobního PC. Při neúspěchu této operace bych přišel ke ztrátě týdenní práce. Podobně v dalším kroku, kde bych se snažil získat zálohu z CD. Při neúspěchu bych mohl ztratit až měsíc práce. Můžu přesto zajít ještě ke specialistovi, který by mohl získat data z poškozeného disku na osobním PC. Zajít ke specialistovi můžu i přes to, že bych získal data z CD, protože ztráta měsíc práce je docela závažný problém. Pokud selžou všechny bezpečnostní prvky, znamená to ztrátu celého dokumentu.

Po sestavení ETA máme dle postupu popsat možné dopady. V našem případě strom obsahuje šest různých dopadů s pěti možnými výsledky. Tímto bych uzavřel první část analýzy. V druhé fázi se přiřadí frekvence inicializační události a pravděpodobnosti vzniku daných uzlů ve stromě. Toto přiřazení umožní vypočítat četnost výsledků a tedy charakterizovat riziko.

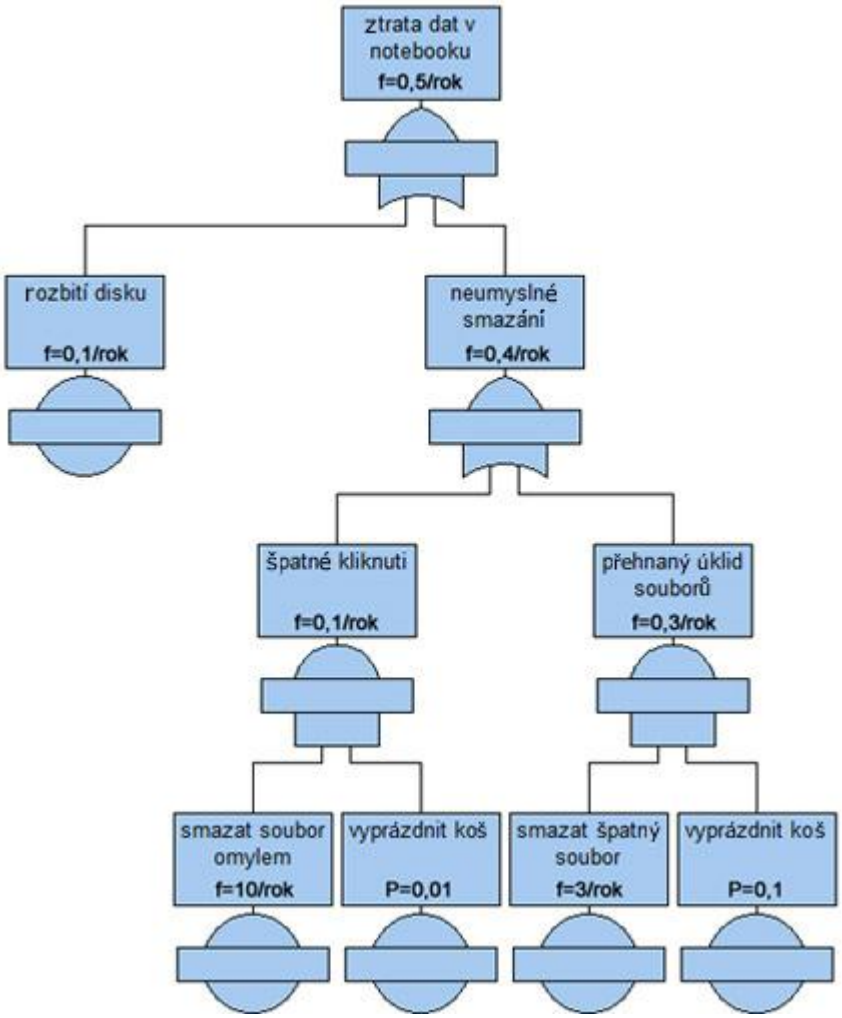


Obrázek 3.5: Ukázka stromu událostí (ETA) – ztráta dat v notebooku

3.2.3 Vyhodnocení ETA v kombinaci s FTA

Při zkombinování ETA a FTA získáme silnou techniku, která nám dokáže ukázat různé dopady určité události a kombinace událostí, které mohou vést k určité události. Pokud bychom chtěli provést kvantitativní analýzu stromu událostí z předchozího příkladu, můžeme subjektivně přiřadit pravděpodobnosti událostem ve stromě. Nicméně existuje ještě detailnější cesta a ta je ve zmíněné kombinaci metody ETA s FTA. FTA má opačný přístup než ETA. Místo toho, abychom u analýzy přemýšleli o událostech, které by mohly nastat, přemýšlíme o tom, jak tyto události mohou nastat. Např. pro inicializační událost „*ztráta dat v notebooku*“ můžeme vytvořit strom chyb a následně ho kvantitativně analyzovat (viz obrázek 3.6). Listy ve stromě chyb znázorňují události, pomocí kterých může dojít ke ztrátě dat v notebooku. Z analýzy vychází, že pravděpodobnost frekvence ztráty dat je 0,5 za rok.

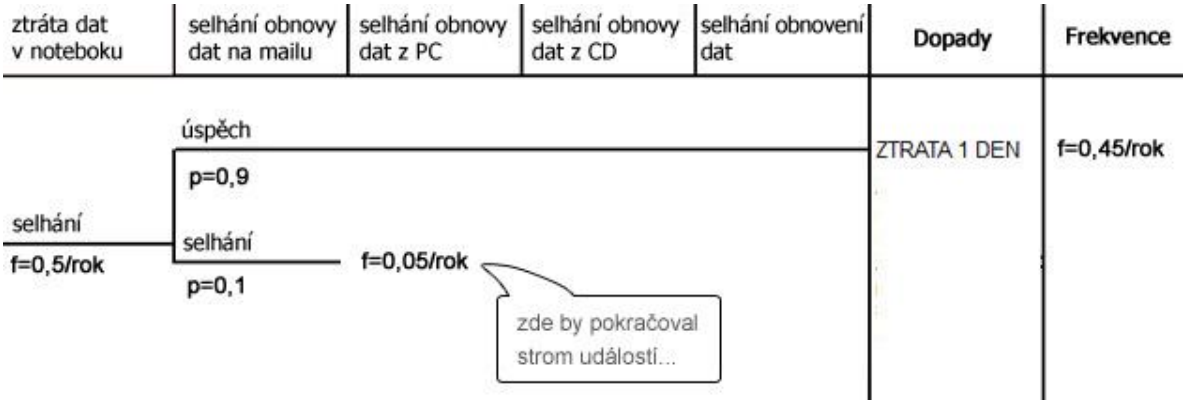
Obdobný strom chyb jako na obrázku 3.6 můžeme udělat pro další úrovně ze stromu událostí. Pravděpodobnost selhaní obnovy dat z mailu by v takovém stromu vyšla 0,1. To znamená, že pravděpodobnost toho, že se nám podaří obnovit zálohu na e-mailu je 90%.



Obrázek 3.6: FTA pro ztráta dat v notebooku [9]

3.2.4 Výpočet frekvence dopadu v ETA

Pokud budeme pokračovat v příkladu z předchozí kapitoly, tak již můžeme zjistit výslednou frekvenci dopadu mimořádné události pod označením „*ztráta 1 den*“. Výsledné frekvence dopadů jsou dány součinem pravděpodobností na jednotlivých uzlech. V našem případě frekvence ztráty jednoho dne se tedy rovná *f = 0,45/rok*. Součet všech výsledných frekvencí dopadů pak musí být roven hodnotě inicializační události a součet všech pravděpodobností v jednotlivých úrovních stromu musí být roven 1 (v tomto případě v první úrovni je 0.9 + 0.1 = 1). Část výpočtu stromu událostí pro zmíněný příklad je na obrázku 3.7.



Obrázek 3.7: Ukázka výpočtu frekvence dopadu v ETA

3.3 Ostatní techniky

V této kapitole bych chtěl poukázat na další možné techniky, které se používají v kvantitativní a kvalitativní analýze rizik. Vzhledem k tomu, že tato práce spočívá v zaměření se na metody FTA a ETA budou tyto ostatní techniky zmíněny pouze ve zkratce.

3.3.1 Metoda What – IF

Tato metoda je založena na otázce přímo z jejího názvu (v překladu co – když?). Respektive pomocí brainstormingu zkoumáme možné neočekávané události na základě otázek typu „Co se stane, když…?“. Definujeme tak nebezpečná místa systému a identifikujeme prvky pro metody FMEA a FTA. Identifikace možných selhání a následků probíhá ve vybrané skupině odporníků dobře seznámených se zkoumaným procesem. Kdokoliv z týmu pak může položit otázku „co se stane, když…?“, která ho zajímá a pracovní tým musí najít odpověď na takto formulovaný dotaz. Postup této metody není tak propracovaný jako v předešlých metodách, proto je podstatně důležité sestavit takový zkušený pracovní tým, který je schopen tvořivého aplikačního přístupu a odhalí intuitivně nebezpečné situace, které se mohou v různých fázích projektu vyskytnout.

Metoda What – If je celkově oblíbená metoda v praxi jelikož není časově příliš náročná. Je však nutno podotknout, že nižší časová náročnost může mít za následek méně systematický intuitivní postup. Nejčastěji se tato metoda používá v průmyslových oblastech při prověřování bezpečnosti studií např. surovin, produktů, skladů, pracovních postupů apod. V nejjednodušší formě má tato metoda výstup v podobě seznamu otázek a odpovědí o šetřeném procesu.

3.3.2 Metoda HAZOP

HAZOP (HAZard and OPerability study - analýza ohrožení a provozuschopnosti) je postup založený na pravděpodobnostním hodnocení ohrožení a z nich plynoucích rizik. Cílem metody je analýza potenciálních nebezpečí pro lidi, životní prostředí, zařízení nebo plán projektu s návrhem scénářů pro jejich kontrolu [10].

Analýza začíná tím, že si definujeme soubor klíčových prvků/slov. Musíme ale brát v úvahu, že se jedná o týmovou expertní multioborovou metodu za využití brainstormingu. Tým se soustřeďuje na posouzení rizika a provozní schopnosti systému. Výhodou HAZOP je, že se používá i pro již existující procesy. Tým si pomocí definovaných klíčových slov pokládá otázky na příčiny typu „co

mohlo způsobit, že …? Na následky je možné požít metodu What – If a pokládat si tedy otázky „co se stane když“. Výstupem metody je závěrečné doporučení, které směřuje ke zlepšení procesu.

3.3.3 Metoda FMEA

FMEA (Failure Mode and Effect Analysis – analýza selhání a jejich dopadu) je postup zaměřený na hledání dopadů a příčin, jež mají značný vliv na bezpečnost a provozování systému/zařízení. Často se FMEA používá pro identifikaci nebezpečí u průmyslových zařízení. Při provádění analýzy se vychází z dokumentů funkčního schématu (technologické schéma, konstrukční výkresová dokumentace apod.). Postup provádění zahrnuje identifikaci každé poruchy u jednotlivých prvků a uvažování sekvence návazných událostí. K těmto poruchám se pak snažíme nalézt příčiny a odhad možných následků. V jednotlivých krocích metody si pokládáme otázky typu: „Jaký je projev poruchy?“, „Jaké jsou možné příčiny poruchy?“, „Jak může být porucha objevena?“, „Jak porucha ovlivní systém?“, „Je tento stav přijatelný?“. U této metody se předpokládá kvantitativní přístup řešení za pomocí počítačové techniky a výpočetních programů s cíleně zaměřenou databází [11].

3.3.4 Metoda HRA

HRA (Human Reliability Analysis - analýza lidské spolehlivosti) je metoda na posuzování vlivu lidského činitele na výskyt nežádoucích událostí a jejich dopadů. Cílem metody HRA je systematicky posoudit a identifikovat potenciální lidské chyby, jejich příčiny a následky. Využíváme zde přístupy mikroergonomické (vztah „člověk-stroj“) a makroergonomické (vztah „člověk-technologie“). Analýza HRA má blízkou vazbu s aktuálně platnými pracovními předpisy především z hlediska bezpečnosti práce. Zabývá se dotazy na charakteristiku prostředí, na dovednosti, schopnosti a znalosti zaměstnanců. Studie a analýza metodou HRA muže být velice pracná, a proto vyžadují patřičné zkušenosti tazatelů. Provádí se jak kvantitativně tak kvalitativně.

3.3.5 Monte Carlo

Tato technika je zaměřena na kvantifikaci rizik, která opakovaně simuluje výsledky modelu a zjišťuje tak statistické rozdělení vypočtených výsledků. Pomocí této metody pak můžeme zjistit pravděpodobnost dokončení projektu. Například víme, že projekt bude dokončen na konci měsíce s pravděpodobností 15 procent, zatímco v polovině příštího měsíce to bude již 55 procent. Můžeme tedy říci, že metoda Monte Carlo nám dokáže předpovědět pravděpodobnost dokončení projektu v určitý den, nebo pravděpodobnost, že náklady na projekt budou nižší nebo zůstanou podle předem odhadnuté hodnoty. Základní kroky při provádění této analýzy jsou [4]:

- Shromáždit nejpravděpodobnější optimistický a pesimistický odhad hodnot veličin v modelu.
- Stanovíme distribuci neboli rozdělení pravděpodobnosti jednotlivých veličin. To znamená, že si určíme například pravděpodobnost toho, že dokončíme projekt mezi optimistickým odhadem a nejpravděpodobnějším odhadem.
- Pro každou z veličin, jako je například odhad doby trvání projektu, vybereme náhodnou hodnotu, která je založena na rozdělení pravděpodobnosti vzniku veličin.
- Nad kombinací těchto hodnot, spustíme deterministickou analýzu neboli jeden průchod modelem.

- Předposlední dva kroky několikrát (zpravidla se provádí 100 až 1000 opakování) zopakujeme, abychom docílili dostatečně věrohodných výsledků rozdělení pravděpodobnosti modelu.

4 Analýza a návrh aplikace

V této kapitole je popsána analýza a návrh mé aplikace pro podporu kvantitativní analýzy rizik zvolenými metodami. Jako metody jsem zvolil metodu stromu chyb (FTA) a metodu stromu událostí (ETA).

Než jsem začal analyzovat požadavky na aplikaci prošel jsem si několik již hotových produktů na trhu, mezi které patří *RAM Commander's FTA software module* od společnosti A.L.D., *FaultTree+* od společnosti Isograph Ltd, *Relex2009 Evaluation* od společnosti PTC, *RiskSpectrum* od společnosti Scandpower. Počet podobných aplikací není nikterak velký a řekl bych, že trh není ještě přesycen. Všechny tyto společnosti se zabývají danou problematikou již několik let a mají pobočky na různých místech světa. Moje aplikace nebude zdaleka tak propracovaná jako výše vypsané, za to se chci ale zaměřit na jednoduchost řešení a software dostupný na internetu jako webová aplikace. Tímto krokem bych učinil moji aplikaci jedinečnou na současném trhu, protože na nic podobného jsem při mém hledání nenašel. Velkou inspiraci díky své jednoduchosti a intuitivnímu ovládání mi byly desktopové aplikace *Relex2009 Evaluation* a *FaultTree+*. Tyto aplikace jsou ovšem distribuovány jako komerční produkt a cena licence za takovouto aplikaci může být pro začínajícího podnikatele vysoká. Navíc aplikace v sobě mají zabudováno nepřeberné množství funkcí, které většina uživatelů z počátku nepotřebuje využívat. Moji aplikaci bych mohl v budoucnu na rozdíl od konkurence nabízet v různých verzích, čímž bych přispěl k oslovení většímu spektru uživatelů.

Následující kapituly popisují jednotlivé fáze analýzy a návrhu, ve kterých jsem popsal očekávané funkce systému, které znázorňuje diagram případu použití v kapitole 4.2. Dále jsem znázornil strukturu databáze pro uložení jednotlivých analýz rizik pomocí ER diagramu a v poslední fázi návrhu je nastíněn předběžný návrh grafického uživatelského rozhraní.

4.1 Neformální specifikace

Cílem projektu je programová aplikace pro podporu kvalitativní a kvantitativní analýzy rizik za pomocí metod FTA a ETA. Tyto dvě metody se dají zkombinovat, čímž získáme silnou techniku, pro modelování rizik. To znamená, že události vytvořené v FTA se budou moci aplikovat pro modelování v ETA. Aplikace by měla být co nejvíce intuitivní a přehledná. Je samozřejmostí, že se tento systém, nebude dát srovnat s konkurenčními systémy, co se týče funkcionality, na druhou stranu bude systém o to jednoduší pro začínající analytiky a bude vyvíjen jako webová aplikace. Ovládání aplikace ve webovém prostředí bude oproti konkurenční desktopové aplikaci možná méně efektivní, za to ale přináší jisté výhody, jako jsou např.:

- zakoupení licence bez nutnosti instalace softwaru na disk
- ukládání analýzy rizik na zálohovaný server
- analýzy rizik budou online dostupné na celém světě 24hod. denně a 7 dní v týdnu
- ochrana dat heslem
- technické zázemí

Výsledkem analýzy stromu chyb bude spočtená pravděpodobnost/frekvence pro každý uzel a pro vrcholovou událost. U analýzy stromu událostí bude výsledkem identifikace a frekvence možných následků inicializační události. Všechny stromy bude možné v aplikaci uložit a později editovat. Pokud bude změněna pravděpodobnost nějaké události ve stromě chyb, která je použita ve stromě událostí, tak se tato změna musí promítnout i ve stromě událostí.

Aplikaci bude moci využívat libovolný uživatel, který se zaregistruje na webové stránce, kde bude aplikace zprovozněna. Při registraci uživatel vyplní své osobní údaje, přihlašovací jméno a heslo. Po přihlášení se uživatel dostane do administrace, kde může spravovat své analýzy rizik.

Vzhledem k tomu, že má být aplikace zprovozněna na webovém serveru budu grafické rozhraní programováno v jazyce XHTML a CSS. Funkčnost aplikace bude realizováno pomocí jazyků PHP a JavaScript. Veškerá data se budou ukládat do databázového serveru MySQL.

4.1.1 Očekávané funkce

Uživatelé systému
Systém bude využívat pouze jeden typ uživatele. Ten bude mít svůj účet chráněný heslem.

Bezpečnost
Vzhledem k závažným analýzám, které může uživatel řešit, musí být systém dostatečně chráněn, aby se do svého účtu přihlásil vždy jen ten, komu účet patří.

Strom chyb
Uživatel bude mít možnost analyzovat rizika pomocí metody FTA. V této části aplikace může uživatel vytvářet nové události a poté je vkládat do stromu chyb, nebo rovnou vytvářet strom chyb z předpřipravených událostí. Vytváření stromu chyb začíná vložením vrcholové události, která se dále rozvíjí ve stromě chyb. Uživatel bude po vytvoření nového stromu vkládat do diagramu stromu chyb pouze události nebo hradla. Hradla budou typu *AND* nebo *OR*. Na výpočet pravděpodobnosti vrcholové události bude použita Esaryho-Proschanova metoda. Výsledkem FTA bude výpočet pravděpodobnosti/frekvence pro každý uzel ve stromě chyb a včetně výpočtu vrcholové události.

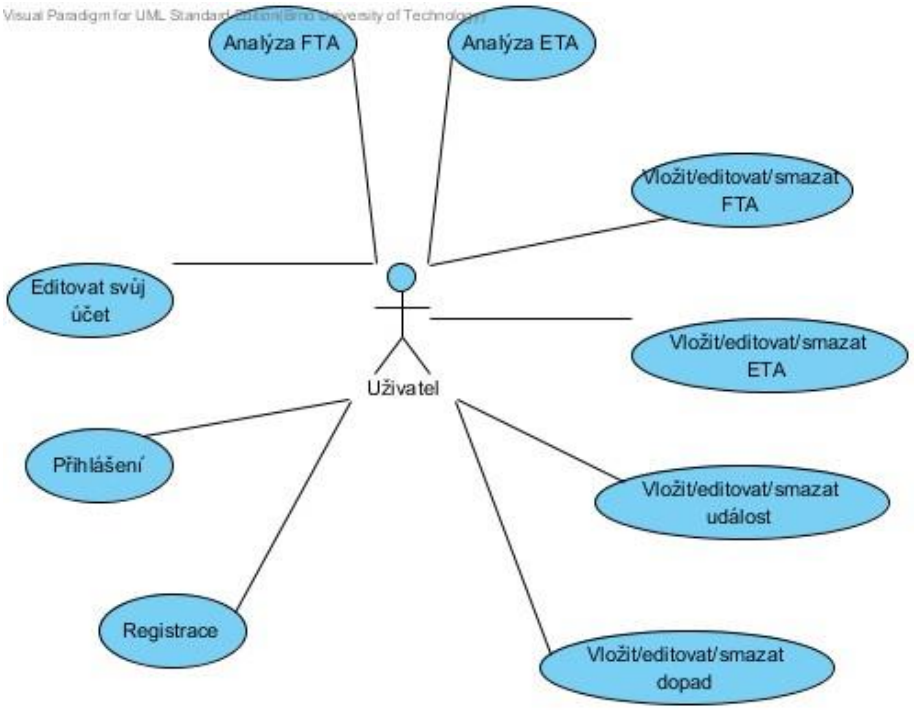
Události
V systému budou události typu *základní událost* a *nerozvíjená událost*. Typy vstupních dat u těchto událostí budou v několika variantách. Jeden z nejvíce používaných typů vstupních dat bude *konstantní pravděpodobnost poruchy*, která má vstupní parametr intenzitu poruchy. Konstantní znamená, že intenzita poruchy dané události bude pevně daná a nebude ji ovlivňovat časová délka analýzy respektive délka života, pro kterou je systém zkoumán, což bude v aplikaci upravovatelná konstanta. Aplikace bude podporovat také pravděpodobnost poruchy události, která bude ovlivněna délkou života systému. Další možný typ vstupních dat bude tzv. *nepohotovost s dobou opravy* (nepohotovost je pravděpodobnostní míra, kdy je zařízení odstaveno nebo není schopné provozu [2]), kde mezi vstupní parametry patří intenzita poruchy a střední doba opravy (více informací o těchto vstupních parametrech bude v kapitole 5.5.1). To znamená, že výsledná pravděpodobnost nepohotovosti události bude ovlivněna průměrným časem opravy události a časovou délkou analýzy. Poslední typ vstupních dat bude *frekvence poruchy*, čili s jakou frekvencí dochází k poruše.

Strom událostí
Další analýzu, kterou bude moci uživatel provádět v systému je ETA. Zde uživatel použije již vytvořené události například z analýzy stromu chyb, což můžou být bezpečnostní událostí (funkce), které mají zabránit nepříznivé inicializační události. Tyto události budou vloženy do jednotlivých úrovní stromu. V každém uzlu se automaticky definuje pravděpodobnost příznivé a nepříznivé větve podle dané vložené události a následně se vypočítá frekvence dopadů nepříznivé události. K těmto dopadům si uživatel může sám nadefinovat jména a popis. Pokud bude potřeba, může si uživatel u uzlu pro nějakou bezpečnostní funkci (událost) nastavit, aby nebyla započítána pro danou větev. To znamená, že uživatel nenastaví úspěšný ani neúspěšný uzel ale nastaví tzv. uzel typu „*NULL*“.

Správa účtu
Uživatel si ve svém účtu může změnit své staré heslo za nové.

4.2 Diagram případu užití

Z neformální specifikace plyne následující diagram případu užití, znázorněný na obrázku 4.1. Tento diagram je modelován pomocí jazyka UML. Diagram představuje chování systému z pohledu uživatele. Vzhledem k tomu, že systém bude používat jen jeden typ uživatele, má i diagram pouze jediného aktéra, který je pojmenován jako *Uživatel*. Uživatel má možnost vkládat nové analýzy stromů chyb a událostí. Tyto dále analyzovat, editovat, popřípadě smazat. Má možnost si před samotným vytvořením stromu chyb, nebo událostí spravovat události a dopady na tyto události. Nedílnou součástí aplikace je vytvoření uživatelského účtu pro přístup do aplikace, který je možné také později editovat.

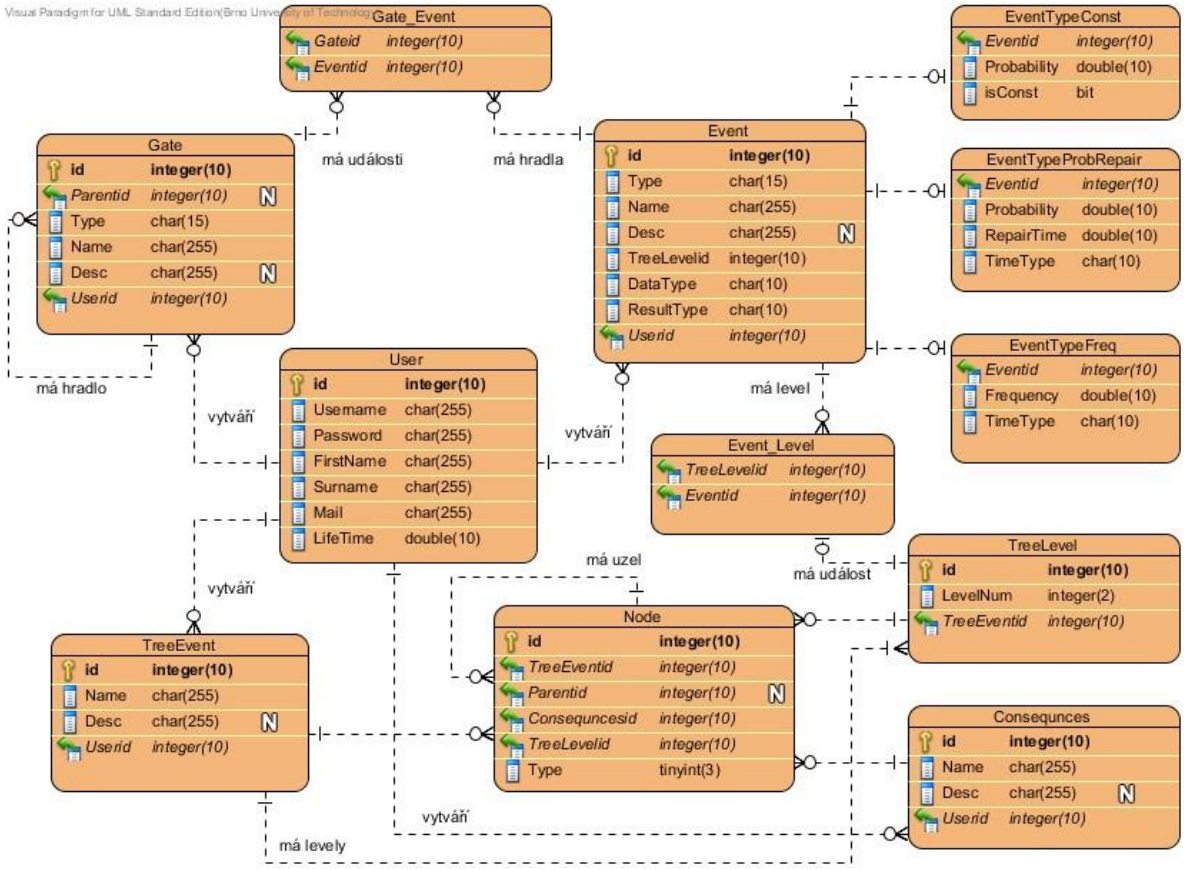


Obrázek 4.1: Diagram případu užití aplikace

4.3 Návrh databáze

Návrh databáze určuje, jak budou data aplikace uložená v databázi. U webových aplikací je tato část procesu návrhu aplikace jednou z nejdůležitějších částí. Proto bylo zapotřebí si pořádně promyslet, jak budou data uložena, aby bylo v budoucnu možné dělat případné aktualizace nebo změny aplikace a nezasáhlo by to nijak razantně doposud navrženou databázi. K tomu, abychom mohli dělat návrh databáze, potřebujeme znát požadavky na systém. Tyto požadavky máme sepsány v kapitole 4.1.

Pro modelování návrhu databáze jsem použil relační model s tabulkovou strukturou. Jedná se o modelování pomocí ER diagramů (Entity Relationship diagram). Je složen z množiny základních objektů (entit) a vztahů (relationship) mezi nimi. Každý objekt má své vlastnosti (atributy), jejíž hodnotu chceme mít v databázi uloženu. Na obrázku 4.2 je znázorněn tento model databáze pro navržený systém na podporu kvantitativní a kvalitativní analýzy rizik s metodami FTA a ETA.



Obrázek 4.2: Návrh databáze aplikace

Základní entitou systému je uživatel (tabulka `user`), který může mít pod svým účtem uložený libovolný počet stromů chyb, stromů událostí, samotných událostí a dopadů inicializačních událostí. Strom chyb bude v databázi uložen jako `Gate` (hradlo), který má `ParentID` rovno `NULL` (nemá žádného předka). Hradla, které nemají žádného předka, jsou zároveň vrcholovou událostí. To znamená, že podle vrcholové události můžeme zobrazit všechny stromy chyb v databázi. Strom

chyb pak bude složen z dalších hradel, nebo událostí (`Event`). Propojení mezi hradly a událostmi zaznamenává entita `Gate_Event`.

Každá událost má svůj typ vstupních dat. Pro každý typ je vytvořena nová entita, s různými atributy. Mezi typy patří konstantní pravděpodobnost poruchy (`EventTypeConst`), nepohotovost s dobou opravy (`EventTypeProbRepair`) a frekvence poruchy, což v databázi odpovídá tabulce (`EventTypeFreq`). U konstantní pravděpodobnosti můžeme nastavit příznak `isConst` na hodnotu `0`. V takovémto nastavení nebude brána hodnota pravděpodobnosti poruchy jako konstantní, ale jako proměnlivá v čase. Časovou konstantu bude možné nastavit u každého uživatele zvlášť v entitě `User` a jeho atributu `LifeTime`. Uživatel bude v systému uložen pod svým uživatelským jménem (`Username`) a bude se moci přihlásit pomocí hesla (`Password`).

Strom událostí bude uložen v tabulce `TreeEvent`. Tento strom se skládá z uzlů (`Node`), který v sobě uchovává id svého rodiče (`ParentId`), abychom byli schopni zjistit složení stromu. Atribut `Type` v tabulce `Node` určí, jestli se jedná o uzel úspěšné větve, neúspěšné větve nebo případně nemá být větev brána do výpočtu (uzel typu `Null`). Každý uzel stromu patří do určité úrovně zanoření stromu (`TreeLevel`). K této úrovni zanoření přiřadíme událost z již uložených událostí, která má určenou svoji pravděpodobnost nebo frekvenci poruchy. K uzlům v nejnižší úrovni stromu bude možnost přiřadit dopady uložené v tabulce (`Consequnces`). Informace o propojení mezí úrovní stromu a událostí patřící do dané úrovně uchovává entita `Event_Level`. O dalších atributech především u entity `Event` se zmíním v kapitole zabývající se implementací systému.

4.4 Návrh grafického rozhraní

Návrh grafického rozhraní definuje, jak bude vypadat prostředí aplikace a kde budou umístěný jednotlivé funkční prvky. Cílem návrhu je, aby aplikace byla co nejvíce intuitivní, přehledná a s co nejmenším počtem potřebných kliků na myš, než se dostane uživatel k požadované funkci.

Vzhledem k tomu, že bude aplikace webová (bude spuštěna v rozhraní internetového prohlížeče), bude vhodné, mít co nejvíce funkčních prvků na hlavní straně, abychom se nemuseli v případě nějaké zanořené funkce, vracet pomocí tlačítka *„zpět“* v prohlížeči. Z těchto důvodů se zobrazí, možnost provádět analýzu FTA nebo ETA, vložit novou událost nebo dopad hned po přihlášení na hlavní stránce aplikace v horní části obrazovky. V pravém horním rohu pak budou o něco méně výrazné ikony pro realizaci odhlášení ze systému a nastavení uživatele. Aby se aplikace co nejvíce přiblížila desktopovým aplikacím, budou se provedené změny načítat pomocí technologie AJAX (více v kapitole 6.2).
Po vybrání libovolné analýzy nebo vkládání dopadu či události zůstane horní část obrazovky stále stejná. Bude se jednat o horní menu aplikace.

V případě výběru analýzy FTA se zobrazí tabulka s uloženými analýzami a možnost přidat novou analýzu. Strom analýzy bude znázorněn v tabulce, kde bude na výběr vložit hradla nebo události. V řádcích tabulky se zobrazí detaily o události nebo hradle, které bude možné editovat. Po navržení stromu chyb se automaticky spustí výpočet analýzy. V tabulce se nám zobrazí pravděpodobnosti poruchy pro jednotlivá hradla a vrcholovou událost.

V případě výběru ETA si při zadávání nové analýzy zvolíme počet úrovní (levelů) stromu. Po potvrzení se vykreslí strom událostí a budeme mít možnost vybrat pro každou úroveň stromu událost. K posledním uzlům stromu bude možné přiřadit dopad z již vytvořených dopadů. Opět se nám automaticky spustí analýza, která spočítá frekvence dopadů po každé změně ve stromně.

Pokud uživatel zvolí v horním menu události, zobrazí se mu v tabulce doposud vytvořené události. Na viditelném místě bude možnost vložit novou událost, což bude realizované formulářovým rozhraní s potřebnými prvky, reprezentující atributy události. Podobně bude fungovat aplikace i při vkládání nového dopadu.

5 Implementace

Implementace systému vychází z předchozí kapitoly a tedy z analýzy a návrhu aplikace. Proto, abychom implementaci mohli udělat nejlépe na poprvé a dobře, musí být nejprve udělán dobře návrh. Přesto i při dobrém návrhu si většinou až u implementace uvědomíme nedostatky návrhu a složitosti některých částí aplikace, které mohou výrazně změnit odhadovaný termín dokončení implementace.

Cílem této kapitoly bylo vytvoření funkční aplikace tak, aby vyhovovala specifikaci v návrhu. Pro splnění těchto cílů jsem potřeboval několik programovacích jazyků a nástrojů pracující s těmito jazyky, které jsou popsány v následujících kapitolách. Dále sou zde také uvedeny některé vybrané implementační části a algoritmy, které se mohly zdát na první pohled jednoduché, ale opak byl pravdou.

5.1 Použité programovací jazyky

Cílem bylo, aby aplikace byla dostupná online, všude na světě a bez nutnosti instalace. Z těchto důvodů jsem musel použít jazyky používané k tvorbě webových aplikací. Vzhledem k rychlosti vývoje technologií právě v této oblasti jsem nemohl zůstat pozadu, a proto jsem pro každou část implementace využil dostupný framework, který je podle mne nejvíce aktuální. Pro návrh rozhraní jsem použil standardní jazyky XHTML a CSS s využitím knihovny YUI od Yahoo. Pro ovládání aplikace na klientské straně jsem použil jazyk JavaScirpt a jeho framework JQuery. Serverovou stranu obsluhuji pomocí jazyku PHP a Nette Frameworku vyvinutého českým vývojářem Davidem Grudlem. Pro uložení dat používám databázový systém MySQL. Ke zjednodušení zápisu SQL příkazů a dalších ulehčujících a bezpečnostních funkcí jsem použil knihovnu Dibi, která je vyvinuta opět Davidem Grudlem. V dalších několika kapitolách jsou popsány zajímavosti jednotlivých vypsaných frameworků.

5.1.1 CSS knihovna YUI

YUI je složeno z několika samostatně použitelných knihoven psaných v jazycích CSS a JavaSCript. Tyto knihovny jsou nástroje pro snadné ladění a také zjednodušují psaní webových aplikací. Pro CSS vytvořilo Yahoo! Inc. tři knihovny. `Rest.css` umožňuje vymazání předdefinovaných stylů prohlížeče, čímž se docílí toho, že ve všech prohlížečích bude shodné výchozí nastavení a předejde se problémům s chybným zobrazením. `Grids.css` slouží pro snadnou editaci rozložení webové stránky. Obsahuje několik nadefinovaných rozložení stránky do rámců. Poslední knihovnou je `fonts.css`, která sjednocuje základní styly fontů pro různé operační systémy a webové prohlížeče [12].

5.1.2 jQuery

JQuery je framework napsaný v jazyce JavaScript, pro ulehčení práce právě s tímto jazykem. Programování v jQuery je na rozdíl od JavaScriptu stručnější, přehlednější a efektivnější. Tento framework oproti obyčejnému zápisu v JavaScriptu obsahuje jednodušší funkce pro provádění

animací, práci s událostmi a hlavně vyhledávaní elementů DOM[1]. Tyto elementy můžeme pomocí jQuery dále editovat, mazat, nebo vytvářet nové, které se automaticky začlení do struktury DOM. Vyhledávají se snadno, pomocí CSS nebo Xpath[2]. Základem je vždy volání funkce jQuery, které lze nahradit znakem dolaru "$". Tato funkce je volána s určitými parametry a podle těchto parametrů je rozhodnuto o způsobu zpracování. Vždy vrátí objekt typu jQuery obsahující kolekci HTML elementů, s kterým je možné dále pracovat.

5.1.3 Nette Framework

Nette Framework je framework využívaný pro tvorbu webových aplikací v jazyce PHP 5. Především se zaměřuje na eliminaci bezpečnostních rizik, znuvupoužitelnost a jednoduchost kódu. V programování používá promyšlený a čistý objektový návrh, událostmi řízené modelování a komponenty díky nimž programátor nemusí psát žádný kód vícekrát. Disponuje bezkonkurenčními ladícími nástroji a exceluje svojí rychlostí. Programátory vede k čistému návrhu aplikace s důrazem na možnou pozdější jednoduchou úpravu. Dokáže výborně rozdělit práci mezi více programátorů a HTML kodérů. Využívá softwarovou architekturu (někdy nazývanou jako návrhový vzor) MVP (Model View Prezenter), který vychází ze vzoru MVC (Model View Controler), o kterém budu psát později v samostatné kapitole. Přes všechny tyhle výhody se navíc jedná o svobodný software[3], nabízený pod licencí GNU GPL[4] a BSD[5] [13]. Aktuálně (duben 2011) je Nette ještě mladý framework ve dvou verzích (verze 0.9 a verze2.0 Alpha 2). Pro tuhle práci jsem využil verzi 2.0 Aplha 2 pro PHP 5.3.

5.1.4 Dibi

Dibi je PHP databázová vrstva, která se snaží ulehčit rutiny, se kterými se programátor běžně setkává. Zjednodušuje zápis SQL příkazů, poskytuje snadný přístup k metodám, i bez globálních proměnných. Má automatickou podporu konvencí („escaping", uvozování identifikátorů), čímž zabráníme útokům typu SQL injection[6] a podobným. Dále Dibi dokáže automaticky formátovat speciální typy jako je například datum nebo řetězec a sjednocuje základní funkce (připojení k databázi, vykonání příkazu, získání výsledku)[14].

5.2 Použité nástroje

Při programování složitějších aplikací se neobejde žádný programátor bez pomocných nástrojů. Je zřejmé, že žádný profesionální software nelze vytvořit komplexně jen v jednom programu, proto znalosti dobrého programátora musí být opravdu rozsáhlé.

Při programování této práce jsem použil vývojové prostředí svobodného softwaru **NetBeans IDE** od společnosti Sun Microsystems. Primárně je tento produkt určen pro vývoj aplikací v jazyce Java, ale podporuje i jiné jazyky jako je např. C++, Ruby a v neposlední řade PHP [15]. Výhodou této aplikace je, že obsahuje velké množství modulů, které toto vývojové prostředí

[1] Vice na: http://cs.wikipedia.org/wiki/Document_Object_Model
[2] http://cs.wikipedia.org/wiki/XPath
[3] http://cs.wikipedia.org/wiki/Svobodn%C3%BD_software
[4] http://nette.org/cs/licence/gpl
[5] http://nette.org/cs/licence/newbsd
[6] http://cs.wikipedia.org/wiki/SQL_injection

rozšiřují a vzhledem k velké uživatelské základně vznikají vývojové moduly pro NetBeans IDE podstatně rychle. Při vývoji této aplikace jsem použil aktuální verzi 6.9.1 (duben 2011) a také nově modul „NetBeans PHP Nette Framework“, který mě podstatně urychlil práci.

Další nedílnou součástí při vývoji webových aplikací je rozšiřující modul **Firebug** pro prohlížeč Mozilla Firefox. Toto rozšíření slouží k ladění a editaci webových stánek. Opět se jedná o svobodný software pod licencí BSD. Velkou výhodou Firebugu je, že obsahuje pokročilou konzolu jazyka JavaScript, díky které můžeme například snadno odladit chyby v aplikaci využívající právě JavaScript. Další obrovskou výhodou je možnost dynamické změny libovolné části webových stránek, díky které můžeme okamžitě vidět například změny vzhledu nebo rozmístění prvků na stránce, aniž bychom přepisovali zdrojový kód.

Při vývoji webových aplikací, které potřebují komunikovat se serverem, si může programátor vybrat ze dvou možností, jak této komunikace docílit. Jednou z možností je, že bude neustále připojen pomocí protokolu FTP (File Transfer Protocol)[7] na server. Při každé změně zdrojového kódu aplikace by ale musel provézt tuto změnu pomocí FTP také na serveru. Druhou možností je varianta, kterou jsem si vybral pro vývoj této aplikace. Jedná se o nainstalování serveru na vlastní počítač. Tato možnost pomůže programátorovi několikanásobně urychlit vývoj, protože není potřeba nová data posílat nikam na vzdálený server, a také není potřeba být připojen k internetu. Veškeré data se berou přímo ze složky na osobním počítači. Pro instalaci serveru jsem použil balíček **XAMPP** vedený pod licencí GPL. Balíček obsahuje instalaci serveru Apache s PHP a MySQL databázi.

Jako poslední nástroj, který jsem využil při vývoji je grafický bitmapový editor **Adobe Photoshop** ve verzi CS5 (12.0) od společnosti Adobe Systems. Tato aplikace je jedna z nejznámějších programů pro úpravu digitálních fotografií, ale také pro vytváření webového designu. Výhodou této aplikace je, že jednotlivé grafické prvky jsou skládány do tzv. vrstev, které je možné zneviditelňovat a mazat aniž bychom narušili ostatní části grafického výtvoru.

Jak již bývá pravidlem, tak dobrý programátor, není nikdy dobrý grafik. Proto je docela obtížné, pokud programátor vyvíjí aplikaci celou sám, aby zvládl všechny její části vývoje na výbornou. Přesto by mělo být dobrým zvykem programátora, navrhnout si grafické uživatelské rozhraní (GUI - Graphical User Interface) v nějakém grafickém editoru, než ho začne programovat. Tímto krokem si můžeme lépe ujasnit přístupy k jednotlivým funkcím systému. Především předejdeme případné zdlouhavé změně, kterou bychom museli provádět v samotném kódu šablony GUI. Já jsem se pokusil pomocí Photoshopu navrhnout GUI a poté jsem si ho připravil do šablony v jazyce XHTML. Při této práci jsem měl možnost vidět design GUI ucelený ještě před jeho kódováním do XHTML, a tak jsem měl možnost, lépe uzpůsobit barvy a rozložení prvků, aby se uživatelovi v aplikaci lépe pracovalo a cítil se v ní dobře a pokliděně.

5.3 Model View Presenter

Při programování jsem využil softwarovou architekturu MVP (Model-View-Presenter), která vychází z více známé softwarové architektury MVC (Model-View-Controller). Jedná se o návrh, který rozděluje aplikaci do tři vrstev (datový model, uživatelské rozhraní a řídící logiku). Tyto tři vrstvy jsou na sobě nezávislé, a tedy modifikace s některou vrstvou nemá vliv (popřípadě má jen minimální) na ostání. Při zamyšlení zjistíme, že jednotlivé kódy webových aplikací skutečně spadají do jedné z těchto kategorií MVP respektive MVC. V praxi se ukázalo, že je dobré, když tyto části

[7] http://cs.wikipedia.org/wiki/File_Transfer_Protocol

od sebe oddělíme do samostatných komponent nebo modelů, čímž dosáhneme přehlednosti kódu a dobré udržovatelnosti především v případě, kdy na aplikaci pracuje více lidí. Nette Framework je právě navrhnut tak, aby odpovídal logice MVP. Pro zjednodušení můžeme říci, že prezenter v Nette je totéž, co controller v jiných frameworcích [13] (např. Zend Framework[8]).

5.3.1 Princip MVP/MVC

Vytváření aplikací s využitím MVC nebo MVP spočívá ve vytvoření tři komponent:

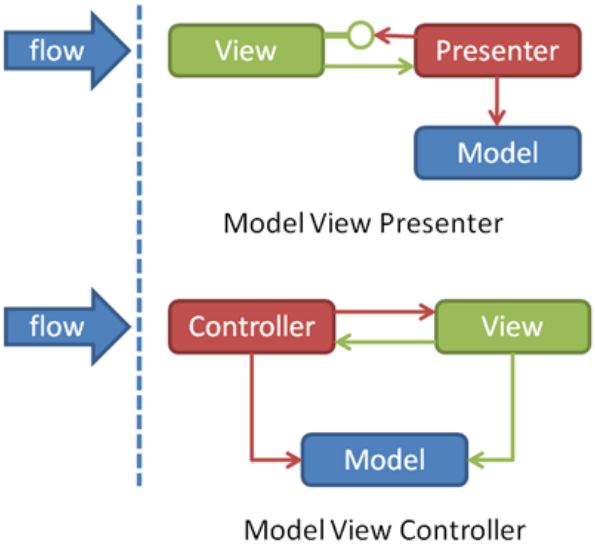
- **Model** – zajišťuje přístup k datům a manipulaci s nimi
- **View** – převádí data reprezentovaná modelem do vhodné podoby reprezentované uživateli
- **Controller/Presenter** – reaguje na události od uživatele a stará se o změny v modelu nebo pohledu

Koncept MVC a MVP bývá v praxi realizován různým způsobem a ve své původní podobě v podstatě není používán. Role a vztahy jednotlivých vrstev jsou často chápany dost volně, a proto se někdy říká, že Nette využívá MVC, i když se jeho architektura podobá MVP. Rozdíly mezi těmito architekturami můžete vidět na obrázku 5.1. Mezi ty nejhlavnější rozdíly patří, že view (pohledu) o modelu nemusí vědět. Presenter komunikuje s modelem a provádí manipulace s pohledovými prvky uživatelského rozhraní. Na druhou stranu MVC má dva případy čtení dat, kdy pohled i controller můžou číst přímo z modelu [16].

Princip činnosti vytvářené aplikace je pak následující:

- Uživatel pracující s aplikací provede nějakou akci (např. klikne na tlačítko)
- Presenter obdrží oznámení o události
- Presenter použije metody z modelu a v případě potřeby ho zaktualizuje podle provedené události z uživatelského rozhraní
- Model zpracuje změněná data (například spočte celkovou pravděpodobnost dopadu) a poskytne je prezenteru.
- Presenter poskytne změněná data pohledu a ten je zobrazí pro uživatele v uživatelském rozhraní. Zde samozřejmě musíme počítat s tím, že u webových aplikací zajistí program obnovení stránek.

[8] http://cs.wikipedia.org/wiki/Zend_Framework



Obrázek 5.1: Rozdíly mezi MVP a MVC [16].

5.4 Adresářová struktura

Přehledná adresářová struktura je základem pro lepší organizaci zdrojových kódů aplikace a snadné pozdější úpravy. Nette Framework nabízí k použití doporučenou adresářovou strukturu, která ale není povinná, přesto jsem ji použil pro vyvíjenou aplikaci. Hlavní částí aplikace jsou tři důležité adresáře: `app`, `libs` a `www`. V adresáři `app` je umístěna vlastní aplikační logika aplikace. Z bezpečnostních důvodů jsou skripty aplikační logiky v odděleném adresáři, který je nedostupný z prohlížeče (složka dostupná z prohlížeče je složka `www`). Každá část aplikační logiky má svůj podadresář, který vychází z modelu MVP. Jedná se o složky `models`, `presenters` a `templates`. V adresáři `presenters` jsou uloženy veškeré řadiče aplikace. V adresáři `models` jsou uloženy jednotlivé modely aplikace (FTA, ETA,…). Poslední adresář `templates` obsahuje šablony, které aplikace používá. Například šablona pro vkládání ETA bude jiná jako pro vkládání FTA. Každá část aplikace má tedy svůj podadresář v adresáři `templates`. V každém takovém podadresáři se již nacházejí šablony pro danou část aplikace. Může jich zde být ale více. Například šablona pro vkládání FTA bude jiná jako pro editaci hradel v FTA, ale obě spadají pod část aplikace FTA a tedy budou obě šablony v adresáři `Fta`.

V adresáři `app` se dále nacházejí dva důležité soubory. Jde o zavádějící soubor celé aplikace, který se jmenuje `bootstrap.php` a má za úkol nastavit prostředí aplikace a spustit ji. Nastavení se skládá z pěti kroků:

- Načtení Nette Frameworku
- Konfigurace prostředí (zapnutí debuggru a načtení konfiguračního souboru `config.ini`)
- Konfigurace aplikace
- Nastavení „routování“ (generování URL, více v [13])
- Spuštění aplikace

Soubor `config.ini` je druhý důležitý soubor v adresáři `app`. Tento soubor složí především k připojení aplikace do databáze.

V adresáři `libs` se nacházejí frameworky Nette a Dibi. Poslední důležitý adresář je `www`, který je veřejně dostupný adresář aplikace. Zde se nacházejí obrázky, CSS soubory, javascriptové skripty a další podobné soubory zobrazované uživatelům nebo vyhledávajícím robotům.

5.5 Implementační části

Implementace aplikace byla rozdělena na několik iterací. V každé iteraci se programovala určitá část aplikace, která je na zbytku aplikace nezávislá. Nejprve jsem si vytvořil databázi podle návrhu databáze v kapitole 4.3. Aplikaci jsem rozdělil na klientskou a administrační část. Tedy na část, která je veřejně dostupná a část, která je dostupná pouze po přihlášení. Přihlášení a registrace byla nedílnou částí klientské aplikace. Díky bezpečnostnímu přihlášení má každý uživatel své analýzy dostupné pouze pro sebe a pod heslem.

Administrační část je rozdělena na FTA, ETA, události a dopady. Každá tato část má svoji vlastní šablonu (template), svůj vlastní presenter a model. Presenter je třída, která dědí od třídy `Presenter` z Nette. Každý presenter v aplikaci se jmenuje `XXXPresenter.php`, kde `XXX` je název presenteru a nachází se v adresáři `app/presenters` (viz kapitola 5.4). Presentery v aplikaci mohou tvořit hierarchii, která je vždy stromová. To znamená, že presenter je buďto třídou `abstrakt` nebo `final`. Každý presenter koná několik akcí, které obsluhují právě jeden požadavek.

Každý presenter má ke každé své akci šablonu. Šablona neboli pohled (view) je zodpovědná jen a pouze za zobrazení dat a jejich prezentaci uživateli. Nikdy se nestará o nic víc. Šablony mají koncovku `phtml` podle frameworku Nette. Šablony mají také svoje povinné pojmenování souborů. Nacházejí se v adresáři `app/templates/YYY`, kde `YYY` znamená název presenteru, ke kterému šablona patří. Šablona samotná má název `XXX.phtml`, kde `XXX` je nahrazeno názvem akce, ke které šablona patří. Uvnitř šablony se nachází kód XHTML, který říká, jak a kde se mají data zobrazit a mimo jiné obsahuje příkazy šablonovacího systému Nette Framework[9]. Šablony se renderují dvoufázově. To znamená, že máme vždy dvě šablony:

- tzv. „šablona layoutu“
- a tzv. „šablona akce“

Šablona layoutu je soubor, který má na začátku názvu zavináč („@“). V mé aplikaci se nacházejí dvě takové šablony `@layout` a `@public`. Šablona `@layout` je výchozí šablona pro backend aplikaci (šablona pro administrační rozhraní) a šablona `@public` je pro frontend aplikaci (šablona pro nepřihlášené uživatele). Šablona layoutu je společná pro více akcí. Především je v ni uložena HTML hlavička a patička webu pro backend respektive frontend.

Šablona akce je taková šablona, o které jsem psal výše, tedy šablona pro presenter, protože každý presenter má svoji šablonu. Kód šablony začíná vždy na příkazu `{block content}`, díky kterému Nette Framework soubor nalezne a vloží jej na patřičné místo do šablony layoutu. V šabloně akce je XHTML kód spolu se šablonovacím systémem Nette a PHP. Slouží k vykreslení výstupu jednotlivých akcí presenteru (např. editování událostí, dopadů, zobrazení stromu chyb, atd.)

[9] http://doc.nette.org/cs/quickstart/hezci-sablony

Proto, abych mohl naplnit šablony daty, vytvořil jsem si několik modelů. Modely jsou třídy napsaná v PHP s využitím knihovny Dibi, které pracují s databází a připravují data pro presenter. Na rozdíl od šablon a presenterů mi zde Nette dával absolutní volnost v pojmenování. Jednotlivé modely jsem pojmenoval tak, aby přibližně odpovídaly třídám, které by byly zakresleny v diagramu tříd. Jedná se o třídy: `Fta`, `Event`, `Eta`, `Conseq`, `UserModel`, což odpovídá objektům pro manipulaci s FTA, událostmi, ETA, dopady a uživateli. Navíc bylo zapotřebí vytvořit třídu `FTAResult`, která pomáhala udržet mezivýsledky při získávání pravděpodobnosti vrcholové události v FTA. O některých složitějších operacích ve třídě FTA a ETA se zmíním v následujících kapitolách. Nejdříve ale upřesním, jaké vstupní události lze v aplikaci vytvořit.

5.5.1 Události

V aplikaci je možné nastavit čtyři druhy vstupních hodnot pro události, které mají dva „datové typy“. Mezi vstupní hodnoty patří:

- Konstantní pravděpodobnost poruchy
- Pravděpodobnost poruchy
- Frekvence poruchy
- Okamžitá nepohotovost

Datový typ události je pouze pomocným typem pro účely aplikace. Nese v sobě informaci o tom, jestli se jedná o událost s pravděpodobností nebo frekvencí. Tato informace je důležitá pro vkládání událostí do stromu chyb i stromu událostí (podrobně bude o tomto psáno v dalších dvou kapitolách). Může se zdát divné, že jsou čtyři možné vstupy, ale jen dva datové typy. Datový typ u konstantní pravděpodobnosti a pravděpodobnosti poruchy je zřejmý. Jedná se o typ události s pravděpodobností. Zde si musíme uvědomit rozdíl mezi pravděpodobností poruchy v čase, která se značí *F(t)* a konstantní pravděpodobností *P*.

Pravděpodobnost poruchy systému odpovídá pravděpodobnosti, že systém selže ne později než v čase *t*. Pro událost s pravděpodobností poruchy v čase, u které víme, že byla funkční v čase $t = 0$, je pravděpodobnost selhání v intervalu $[0, t]$ rovna *F(t)*, což je ve vztahu k “hustotě pravděpodobnosti“ podle rovnice [2]:

$$F(t) = \int_0^t f(t')dt' \tag{1}$$

V kvantitativní analýze je předpokládáno, že každá událost je na začátku provozu bez poruchy a za nějakou dobu musí selhat. Z tohoto vyplívají dvě rovnice během života události:

$$F(0) = 0 \tag{2}$$

$$F(\infty) = 1 \tag{3}$$

Rovnice (1) pro *F(t)* může být podle těchto okrajových podmínek derivován jako:

$$\frac{dF(t)}{dt} = f(t) \tag{4}$$

Pro výpočet pravděpodobnosti poruchy dále potřebujeme znát tzv. intenzitu poruch systému. Ta je definována jako:

$$\Delta(t) = \frac{f(t)dt}{dt(1 - F(t)} \tag{5}$$

V definici intenzity poruchy (5) je *f(t)dt* pravděpodobnost poruchy v infinitezimálním intervalu *dt* podle času *t* a *Δ(t)dt* je pravděpodobnost, že systém neselže dříve, jak v čase *t*. Pokud spojíme (4) a (5) dostaneme následující diferenciální rovnici:

$$\Delta(t) = \frac{\frac{dF(t)}{dt}}{1 - F(t)} = \frac{-d\ ln(1 - F(t))}{dt} \tag{6}$$

Po integraci (6) získáme rovnici:

$$1 - F(t) = e^{-\int_0^\infty \Delta(t')dt'} \equiv R(t) \tag{7}$$

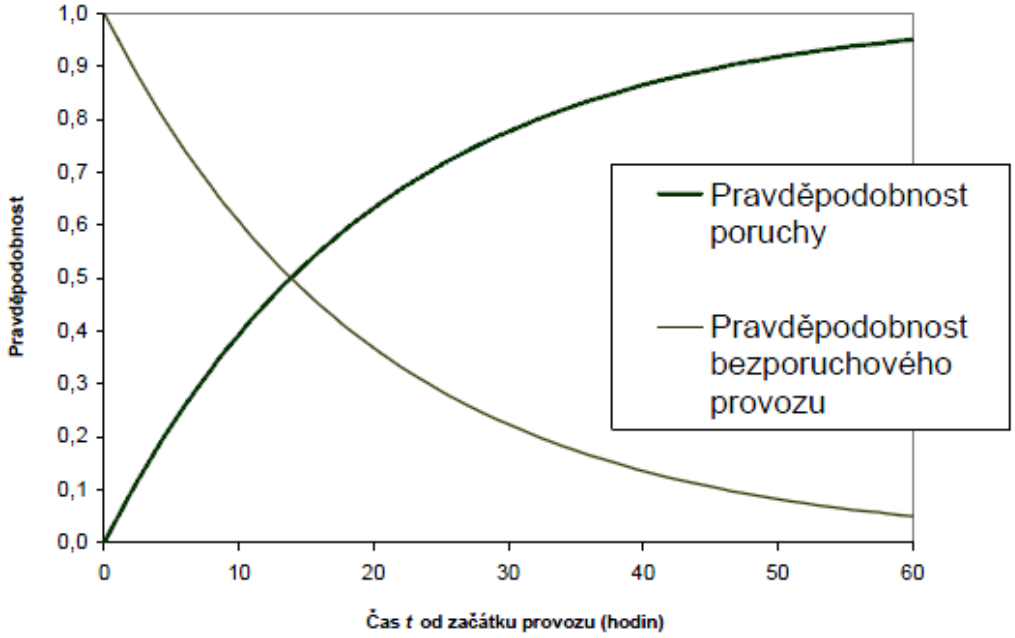
R(t) v kvantitativní analýze udává pravděpodobnost bezporuchového provozu. Tato pravděpodobnostní míra nám tedy říká, po jakou dobu se daná událost nevyskytne v měřeném systému pro určité časové období *t*. Je možné ji vyčíslit takto:

$$R(t) = e^{-\int_0^\infty \Delta(t')dt'} = e^{-\Delta t} \tag{8}$$

Doplněk k pravděpodobnosti bezporuchovosti systému *R(t)* je *F(t)*, a tedy $F(t) = 1 - e^{-\Delta t}$. Tuto rovnici jsem také využil v aplikaci pro počítání výsledné pravděpodobnosti poruchy v čase *t*. Obrázek 5.2 ukazuje funkci pravděpodobnosti bezporuchového provozu a pravděpodobnosti poruchy s intenzitou poruch $\Delta = 0.05$h/hod. Z obrázku jde vidět, jak se pravděpodobnost bezporuchového provozu s rostoucím časem snižuje od 1 až téměř k nule. Pravděpodobnost poruchy je doplněk *R(t)*, a proto má opačný průběh. S rostoucím časem se zvyšuje pravděpodobnost dané události resp. systému.

Co se týče konstantní pravděpodobnosti, tak ta se v čase nemění. Tyto události se dějí pravidelně a není možné je v čase ovlivnit. Proto zde není potřeba žádného složitého výpočtu, ale rovnou se pracuje s danou hodnotou intenzity poruchy.

U událostí pracující s frekvencí v daném čase, je datový typ samozřejmě typu frekvence. Událost, která má zadanou frekvenci poruchy se pravidelně děje v nějakém intervalu (např. hodina, den nebo rok) a stejně jako u konstantní pravděpodobnosti, i zde se pracuje přímo s danou hodnotou frekvence jak u výpočtu v FTA, tak u ETA.



Obrázek 5.3: Graf pravděpodobnosti bezporuchového provozu a pravděpodobnosti poruchy s intenzitou poruch Δ = 0,05/hod [2].

U funkce okamžité nepohotovosti systému je datový typ mírně zavádějící. Nepohotovost je definována jako pravděpodobnost toho, že systém je ve stavu neschopném plnit v daném časovém okamžiku požadovanou funkci. Zde je k výpočtu potřeba znát MTTR (Mean Time To Repair) a intenzitu poruch systému. MTTR je zkratka pro metriku, která znamená tj. průměrnou dobu odstávky systému vlivem oprav, neboli průměrná doba nutná k opravě/výměně měřeného subjektu. Tato doba bývá nejčastěji udávána v hodinách. Čím je tato doba menší, tím dříve dojde k opravě a tedy je měřený subjekt spolehlivější [18]. Výsledná hodnota funkce nepohotovosti je spočtena podle rovnice (9). Výstupem této funkce je pravděpodobnost nepohotovosti události a tedy i datový typ pro tuto událost bude typ s pravděpodobností. Z toho vyplívá, že stačí mít v aplikaci jen dva datové typy. Rovnice pro spočtení nepohotovosti systému je následující:

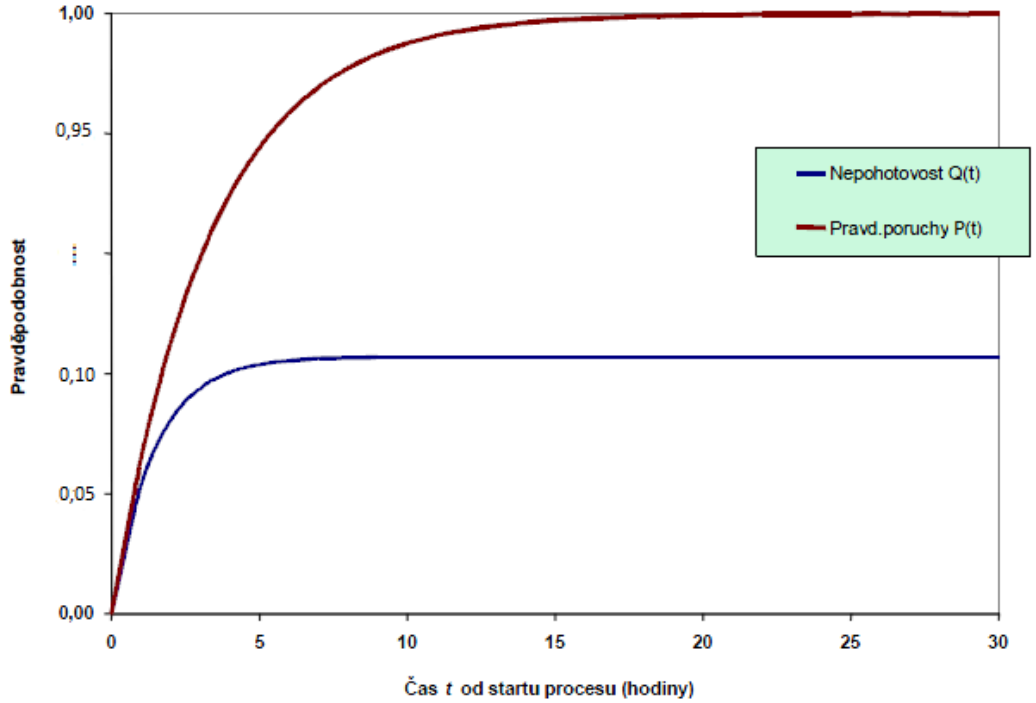
$$Q(t) = \frac{\Delta}{\Delta + \mu^{-1}}\left(1 - e^{-(\Delta+\mu^{-1})t}\right) => t \to \infty => Q(t) = \frac{\Delta}{\Delta + \mu^{-1}} \tag{9}$$

Zde je Δ je intenzita poruch systému, a μ je „střední doma do opravy“ (MTTR). Pokud je událost brána jako neopravitelná, nepohotovost přechází v pravděpodobnost poruchy tj. $Q(t; \mu=0) = F(t)$ [2]. V takovém případě bychom použili místo nepohotovosti pravděpodobnost poruchy. Důležitým rozdílem mezi nepohotovostí $Q(t)$ a pravděpodobností poruchy $F(t)$ je jejich chování pro delší časovou periodu. S delším časem t se blíží $F(t)$ hodnotě 1, zatímco $Q(t)$ dosahuje ustálené hodnoty $\frac{\Delta}{\Delta+\mu^{-1}}$. Obrázek 5.2 znázorňuje tuto vlastnost pro náhodně zvolená čísla Δ a μ.

Vzhledem k tomu, že některé typy vstupních událostí jsou závislé na čase, je potřeba v aplikaci nastavit časovou hodnotu t, uvádějící dobu, pro kterou má být událost zkoumána. Výchozí hodnota je nastaveno t =1000 časových jednotek, kterou je možné měnit.

Při analýze stromu chyb se často muže stát, že víme, z jakých událostí se strom bude skládat, ale nevíme pravděpodobnostní hodnoty událostí. V takovém případě se používá událost

nerozvíjená, která je v aplikaci také zahrnuta. V opačném případě použijeme v aplikaci událost základní.



Obrázek 5.2: Příklad pravděpodobnosti poruchy a nepohotovosti systému pro $\Delta = 0.3/hod$ a $\mu = 0.4/hod$ [2].

5.5.2 Implementace FTA

Analýza stromu chyb, jak již bylo řečeno v kapitole 3.1, se zabývá identifikací a analýzou událostí, které mohou způsobit výskyt vrcholové události. Ve spoustě odborných publikacích zabývajících se FTA, jsou výskyty vrcholové události počítány zjednodušeně a to tak, že pokud jsou události spojeny hradlem AND, tak se pravděpodobnosti násobí a pokud jsou spojeny hradlem OR, tak se sčítají. Takto zjednodušeně samozřejmě nepočítá, žádný software provádějící tento druh analýzy. U většiny softwarů se využívá Esarycho-Proschanova metoda (EPM), kterou jsem vysvětlil v kapitole 3.1.5.2. Tuto metodu jsem se rozhodl uplatnit i v mém algoritmu výpočtu vrcholové události. EPM využívá při výpočtu minimální kritické řezy. Tyto řezy jsou množiny událostí, které při současném výskytu způsobí vrcholovou událost. Zjednodušeně se dá říci, že pokud budou události spojeny hradlem OR, nastane minimální kritický řez v tomto hradle při výskytu libovolné události, která je spojena s tímto hradlem. Opačně, pokud máme hradlo AND, nastane minimální kritický řez v tomto hradle pouze, pokud nastanou všechny události spojeny s tímto hradlem. Při dodržování těchto pravidel, lze vytvořit tabulku, pomocí které lze odvodit algoritmus výpočtu. Tato tabulka je znázorněna v tabulce 5.1. Tabulka také znázorňuje další pravidla sestavení stromu chyb, které často porušují začínající analytici v analýze stromu chyb.

5.5.2.1 Výpočet a pravidla ve stromě chyb

Pravidla, které se musí při sestavení a výpočtu FTA dodržovat jsou určeny podle druhu hradla a typu vstupních dat v události. Pokud máme hradlo *OR* a vstupní události mají hodnotu vstupních dat zadanou jako pravděpodobnost, bude algoritmus výpočtu postupovat podle rovnice (10), která je rovněž v tabulce 5.1. Znak *P* značí pravděpodobnost poruchy a znak *F* značí frekvenci poruchy dané události. Pokud bychom tedy měli hradlo *OR*, které by spojovalo dvě události, kde první událost by měla pravděpodobnost výskytu 0.4 a druhá událost pravděpodobnost výskytu 0.2, byla výsledná pravděpodobnost hradla *OR* spočtena dle rovnice (10) jako *P = 1 - (1 - 0.4) * (1 – 0.2) = 0.52*.

$$P(A\ OR\ B) = 1 - (1 - P_A) \cdot (1 - P_B) \tag{10}$$

Pokud budeme mít hradlo *OR* a vstupní události mají hodnotu vstupních dat zadanou jako frekvence, bude algoritmus počítat výslednou frekvenci opět podle tabulky 5.1, tedy sečte všechny frekvence pod daným hradlem.

Častou chybou začínajících analytiků bývá vložení dvou a více událostí s různými vstupními datovými typy do hradla *OR*. To znamená, že není dovoleno vkládat událost typu pravděpodobnost a frekvence pod jedno hradlo *OR*. Podobně jako není dovoleno vkládání dvou a více událostí typu frekvence pod jedno hradlo *AND*. Tyto začátečnické chyby jsou v mé aplikaci zachyceny a vizuálně znázorněny uživatelovi červeným podbarvením události a hradla. O tomto chování aplikace se zmíním ještě později v kapitole zabývající se ovládáním aplikace.

Hradlo	Párování vstupů	Výpočet pro výstup
OR	P_A OR P_B	$P(A \text{ OR } B) = 1 - (1 - P_A)\cdot(1 - P_B) = P_A + P_B - P_A \cdot P_B \cong P_A + P_B$
	F_A OR F_B	$F(A \text{ OR } B) = F_A + F_B$
	P_A OR F_B	Není dovoleno
AND	P_A AND P_B	$P(A \text{ AND } B) = P_A \cdot P_B$
	F_A AND F_B	Neobvyklé párování, nutno převést na F_A AND P_B
	F_A AND P_B	$F(A \text{ AND } B) = F_A \cdot P_B$

Tabulka 5.1:Pravidla pro výpočet FTA [2]

Vzhledem k tomu, že strom chyb může mít několik zanořených úrovní pomocí hradel, musíme přenášet datový typ i mezi hradly. Hradlo získává datový typ podle událostí pod daným hradlem. Hradlo s datovým typem pravděpodobnost, se změní na typ frekvence, jakmile načte událost se vstupní hodnotou zadanou jako frekvence. Z toho také vyplývá, že datový typ vrcholové události bude frekvence. Pokud se událost s frekvencí ve stromě nikde neobjeví, bude vrcholová událost vypočtena jako pravděpodobnost poruchy. Právě pro tuhle kontrolu datového typu jsem využil instanci třídy `FtaResult`. Algoritmus výpočtu neustále kontroluje datový typ aktuální události (aktuálního objektu `FtaResult`) a následující události (následního objektu `FtaResult`). Pokud se jednou objeví typ frekvence, tak se tento typ posouvá až do vrcholové události resp. hradla. V průběhu celého procesu se samozřejmě kontroluje, jestli nějaké hradlo nezískalo typ frekvence a

není zanořeno pod dalším hradlem s nějakou základní událostí. Například pokud by bylo ve stromě hradlo *OR*, které spojuje hradlo typu frekvence s událostí typu pravděpodobnost, tak by se v takovém případě opět vyskytla ve stromě chyba, která by byla zobrazena v aplikaci červeným podbarvením hradla OR.

5.5.2.2 Vykreslení stromu chyb

Vykreslení a uložení stromu probíhá v cyklu tak, že se nejdříve načtou všechny události pod daným hradlem, které má `parentId` rovno `NULL`. Tyto hradla jsou v aplikaci brána jako vrcholové události a rovnou jsou považovány jako jednotlivé uložené analýzy stromu chyb. Po načtených událostech ve vrcholovém hradle se načtou všechny hradla, které mají `parentId` rovno `id` hradlu vrcholovému. Jednoduše se dá říci, že se načítají všechny sub hradla vrcholového hradla. Na tyto sub hradla se volá stejná funkce jako na vrcholové hradlo. Tedy opět se načtou všechny události pod daným sub hradlem a pak všechny sub hradla. Tento cyklus se opakuje, dokud nedojde k vykreslení celého stromu. V tomto procesu se rovnou vypočte pravděpodobnost resp. frekvence vrcholové události případně všech hradel ve stromě.

5.5.3 Implementace ETA

Analýza stromu událostí identifikuje možné následky inicializační události, která může být například i vrcholová událost analyzována ve stromě chyb. Více o analýze událostí je popsáno v kapitole 3.2. Na rozdíl od stromu chyb má strom událostí vždy z jednoho uzlu dvě další větve. I když je počítání frekvence následků jednodušší, vykreslení a uložení stromu událostí pomocí internetového prohlížeče je o to složitější.

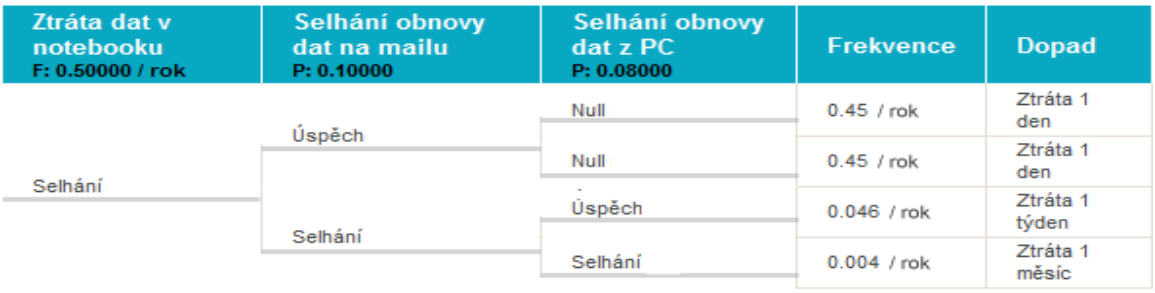
5.5.3.1 Vytvoření stromu událostí

Proces vytvoření stromu začíná tak, že si uživatel nejdříve zvolí počet úrovní (bezpečnostních opatření, nebo možných následujících poruch v systému) stromu. Do databáze se uloží nová analýza stromu událostí, vytvoří se požadovaný počet úrovní v tabulce `TreeLevel` a rekurzivně se vytvoří jednotlivé úspěšné a neúspěšné uzly pro dané `id` levelu. Do takto vytvořeného stromu, je nyní možné vkládat události pro jednotlivé úrovně. Pokud bychom například do nějaké úrovně vložili událost s pravděpodobností poruchy 0.3, budou mít všechny úspěšné uzly patřící do dané úrovně pravděpodobnost 0.7 a všechny neúspěšné uzly pravděpodobnost 0.3. To znamená, že s pravděpodobností 0.7 se podaří riziko (inicializační událost) zastavit pro danou úroveň a s pravděpodobností 0.3 se např. bezpečnostní opatření systému prolomí a pokračuje se do další úrovně, respektive na další bezpečnostní opatření s novou pravděpodobností.

5.5.3.2 Výpočet frekvence dopadů

Protože pro některé uzly nemusí zbylé bezpečnostní opatření již platit (např. obrázek 5.3, který znázorňuje zkrácený strom z obrázku 3.7 v kapitole 3.2.4, kde se podařilo získat zálohované data z mailu s pravděpodobností 0.9, a tedy ostatní bezpečnostní opatření již nepřipadají v potaz), může uživatel nastavit na vybrané uzly, aby se dané bezpečnostní opatření nezapočítalo do výsledného dopadu. V algoritmu výpočtu se pak kontroluje, jaký má uzel typ. Pokud má typ `NULL` (v databázi je hodnota typu rovna 0), znamená to, že doposud spočtená frekvence dopadu bude násobena číslem 1, a tedy se neprovede žádná změna. Pokud má uzel typ *Úspěch,* provede se vynásobení hodnoty doposud spočtené frekvence dopadu s hodnotou pravděpodobnosti bezporuchovosti události. Pokud

má uzel typ *Selhání*, provede se vynásobení hodnoty doposud spočtené frekvence dopadu s hodnotou pravděpodobnosti poruchy události. Výsledná hodnota každého dopadu je pak dána součinem všech hodnot uzlů pro danou větev.



Obrázek 5.3: Ukázka analýzy stromu událostí z aplikace

5.5.3.3 Vykreslení stromu událostí

Proces vykreslení stromu se provádí rekurzivním vykreslováním části šablony. Vykreslování části šablony, což je označený blok HTML, mi umožnil Nette Framework. Díky Nette jsem schopen označit libovolnou část šablony tím, že ji uzavřu do bloku pomocí makra `{block}`. Části šablony označené jako bloky jsou vyjmuty ze svých původních míst v šabloně a vloženy na místo, které je označeno makrem `{include}`. Při deklaraci bloku do něj automaticky přecházejí všechny lokální i globální proměnné. Díky tomu jsem si mohl do bloku poslat objekt s vybraným stromem událostí a postupně ho vykreslit. Blok v tomto případě znamená obsah řádku tabulky označený tagy `<tr></tr>`. Tělo bloku lze zjednodušeně zapsat následovně:

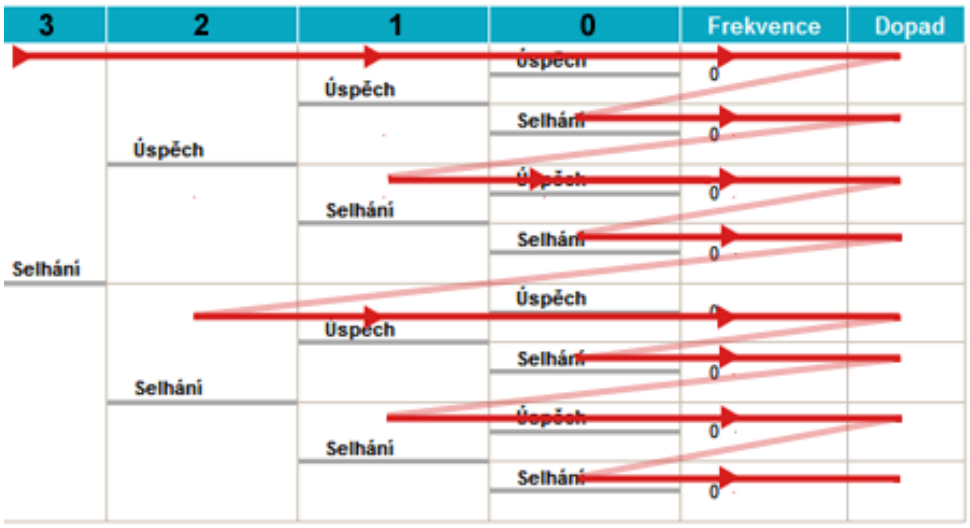
```
{block #printLevel}
     <td rowspan="{=pow(2, $node->level)}">
           {if ($node->type == 1)}Úspěch
                 {elseif ($node->type == 0)}Selhání
                 {elseif ($node->type == 2)}Null
            {/if}

            {if $node->level != 0}
                        {include #printLevel 'node'=>$node->childs[0]}
                        {include #printLevel 'node'=>$node->childs[1]}
                 {else}
                        <td>{$node->result}</td>
                        <td>{$node->cname}</td>
                 </tr>
                 <tr>
            {/if}
{/block}
```

Volání bloku je pak na správném místě v šabloně zapsáno takto:

```
…
<tbody>
      <tr>
            {include #printLevel 'node'=>$tree}
      </tr>
</tbody>
…
```

Obsah bloku pojmenovaného jako `printLevel` vypisuje jednotlivé uzly stromu událostí, které jsou vypsány jako buňky tabulky. Pomocí atributu `rowspan`, který nám určuje přesah buňky na *n* dalších řádků u tagu `<td>`, jsem schopen vykreslit tabulku, která bude mít vzhled, připomínající binární strom. Hodnota atributu `rowspan` je v algoritmu vykreslení stromu počítána jako *x-tá* mocnina čísla dvě, kde *x* je počet úrovní (levelů) v daném stromě číslovaný od 0. Pokud budeme mít například strom s počtem úrovní *4*, bude první buňka tabulky přesahovat na 8 dalších řádků (3^2 = 8). Následně se do buňky vypíše, jestli se jedná o úspěšný, neúspěšný nebo ignorovaný uzel a pokud nejsme na konci řádku tabulky (level je různý od 0), znovu vložíme blok do aktuálního bloku s parametrem obsahující vrchní a poté spodní větev. Pokud algoritmus dojde, až na konec řádku tabulky, vypíše výsledek obsahující frekvenci dopadu pro danou větev a za ní do nové buňky vypíše jméno dopadu přiřazeného k dané větvi. Průběh vykreslování celé tabulky znázorňující strom událostí je v obrázku 5.4 zobrazen červenou linkou. Na obrázku jde vidět, že první je vykreslena buňka zleva (level 3) přes všech osm řádků. Vypíše se typ uzlu jako *Selhání*, protože počáteční uzel je vždy takovéhoto typu a nelze ho měnit. Nyní se zanoříme do vrchní úspěšné větve v levelu 2 a pak ještě dvakrát v levelu 1 a 0, dokud nenarazíme na konec tabulky. Vypíše se výsledek obsahující frekvenci dopadu pro danou větev a za ní aktuální jméno dopadu. Pokračuje se tím, že algoritmus vykreslí spodní větev pro řádek 2 v levelu 0 a hned narazí na konec tabulky. Opět se vypíše frekvence a dopad. Nyní se vynoří z rekurze v levelu 1, vykreslí spodní větev *Selhání* pro řádek 2 a pokračuje vrchní větví v levelu 0. Tam narazí na konec tabulky, takže hned pokračuje spodní větví v řádku 4. Tato spodní větev nemá žádného potomka, takže se vynoří v rekurzi až v levelu 2 a tam pokračujeme v neúspěšné větvi obdobně jako doposud.



Obrázek 5.4: Vykreslení stromu událostí pomocí tabulky

5.5.3.4 Pravidla pro vkládání událostí do stromu událostí

Jedním z důležitých pravidel, které nese vkládání událostí do stromu je vložení počáteční události do nejlevější úrovně. Tato událost může být pouze typu frekvence. Ostatní události ve stromě musí být typu pravděpodobnost. Některé konkurenční programy tato pravidla neřeší (rovnou provedou výpočet špatně a nijak neupozorní na chybu) a začínající analytici s tím mohou mít problémy. Přizpůsobil jsem aplikaci tak, aby nebylo možné vložit událost s pravděpodobností jako inicializační a vložit událost s frekvencí do ostatních úrovní stromu. Události se do stromu vkládají tažením myši. Tento způsob funguje tak, že určíme, jaký HTML prvek je možné táhnout myší a do jakého prvku tento prvek můžeme vložit. Při vykreslování událostí je nastaven atribut `class` na hodnotu `freq` nebo `prob` u prvků znázorňující události. Hodnota atributu je převzata z databáze ze sloupce `resulttype`. Tento sloupec má hodnoty pouze `freq` nebo `prob` podle typu události.

Místo, kam se události vkládají, jsou buňky v tabulce znázorňující jednotlivé úrovně stromu. Počet sloupců tabulky je tedy podle počtu úrovní. Aby bylo možné vložit událost s frekvencí jen do první buňky v prvním sloupci zleva, využil jsem Nette makro `foreach`. `Foreach` makro se chová jako běžný cyklus `foreach` v php, jen má navíc několik rozšíření. Uvnitř cyklu je inicializována proměnná `$iterator`, díky které lze zjistit údaje o probíhajícím cyklu. Proměnná `$iterator` má několik metod, mezi které patří metoda `isFirst()`, určující, zda se prochází cyklem poprvé. V této podmínce je nastaven atribut `class` v šabloně u buňky označující úroveň stromu, kde se vkládá inicializační událost na hodnotu `level-freq`. U ostatních buněk označující zbylé úrovně, je nastaven atribut `class` na hodnotu `level-prob`, čímž je dosáhnuto zabezpečeného vkládání událostí.

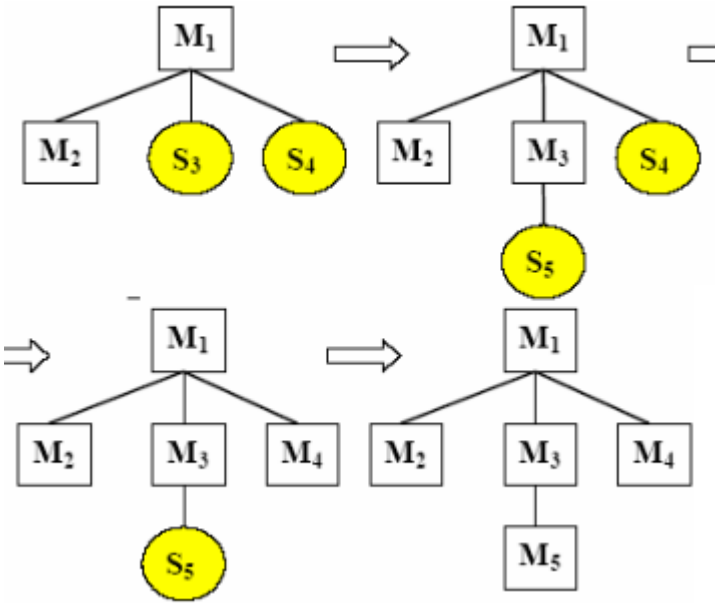
Aplikace by byla schopna vykreslit neomezeně velké stromy. Přesto jsem se ji rozhodl omezit na možnost vykreslení stromu s maximálně sedmi úrovněmi. Větší stromy nejsou v praxi doporučovány, protože se v nich ztrácí přehlednost. Pokud se jedná o složitější analýzu, tak bývá rozkreslena do více stromů. Takto složité analýzy ale nebývají časté. Například atomové elektrárny mají většinou pět bezpečnostních opatření, respektive událostí, které mohou nastat po inicializační události. Větší bezpečnostní opatření se běžně nepoužívají.

5.5.4 Průběh testování shora dolů

Implementace rozsáhlejší aplikace bývá typicky sestavena z řady podsystémů nebo komponent, z nichž některé mohou být implementovány souběžně, jiné postupně. V průběhu vývoje, ale někdy potřebujeme sestavit z několika implementovaných komponent celý systém nebo alespoň nějakou jeho větší část, abychom mohli provádět průběžné testováni správnosti aplikace. Je totiž zřejmé, že i když budeme v rámci implementace testovat jednotlivé komponenty a odstraníme chyby, není to zárukou, že po spojení komponent do celkového systému bude vše v pořádku.

Při vývoji aplikace jsem často využíval testování shora dolů. Znamená to, že jsem měl sestavenou komponentu na vyšší úrovni, která používala komponentu, nebo část aplikace na nižší úrovni viz obrázek 5.2. Význam vazeb komponent v obrázku znázorňuje, že komponenta M1 používá komponentu M2, M3 a M4. Komponenta M3 navíc používá M5. Vzniká zde tedy problém, že testovaná část, může používat komponenty, které ještě nejsou implementovány. Například při testování komponenty M1 a M2 nejsou ještě k dispozici komponenty M3 a M4, které jsou potřebné pro test M1. Aby mohl test proběhnout, musí být něčím nahrazený. Těmto náhradám se říká náhražky (stubs) a jsou v obrázku poznačeny jako S3 až S5. Náhražky mohou být realizovány pouhým zobrazením nějaké zprávy, nebo ručním zadáním hodnot [17]. Například při vývoji stromu

událostí je potřeba, aby byla hotova část aplikace realizující přidávání událostí do stromu. Proto jsem musel vytvořit náhražku a ručně zadávat data do databáze. Obdobný postup probíhal i při testování dopadů. Při výpočtu stromu chyb jsem musel ručně vyplnit tabulku určující, které události jsou umístěny v hradlech, jelikož jsem neměl vytvořenou část aplikace, přidávající události do hradel, ale potřeboval jsem testovat výpočet pravděpodobnosti výskytu vrcholové události.



Obrázek 5.2: Průběh testování shora dolů [17]

6 Funkce systému

Tato kapitola detailně popisuje funkce celého systém. Téměř by se tato kapitola dala přirovnat k jakémusi popisu ovládání aplikace. Tato část se může zdát jako bezúčelná, ale i já jsem měl problém zorientovat se a pochopit konkurenční aplikace, i když jsem průběhům analýz rozuměl. Naopak pokud někdo nemá alespoň základní znalosti v FTA a ETA, stane se pro něj aplikace nepochopitelná. Samotná aplikace totiž nikoho nenaučí provádět žádnou z analýz, ale pouze pomůže analytikovy při provádění těchto analýz.

6.1 Registrace a autentizace uživatele

Vzhledem k tomu, že aplikace je webová a tedy dostupná pro libovolný počet uživatelů z jednoho místa a online 24 hodin denně, bylo zapotřebí, nějak umožnit uživatelům pracovat s aplikací, aniž by se ovlivňovali s ostatními uživateli. Tento problém řeší registrace uživatelů. Každý uživatel si může vytvořit libovolný počet účtů v tzv. části frontend aplikace (část aplikace pro nepřihlášené uživatele) a v nich si spravovat své analýzy, ke kterým nemá nikdo jiný přístup. Při vytváření účtů je heslo zakódováno algoritmem SHA-1 (Secure Hash Algorithm). Tento algoritmus se řadí mezi hašovací funkce, které vytváří ze vstupních dat (hesla zadaného od uživatele) otisk (hash) fixní délky o velikosti 160 bitů. To znamená, že z libovolného množství vstupních dat vytvoří hašovací funkce stejně dlouhý výstup. Malou změnou vstupních dat dosáhne velké změny na výstupu a je prakticky nemožné zjistit původní text zprávy (hesla) [19]. Jedním možným útokem na prolomení hesla je slovníkový útok. K tomuto typu útoku útočník používá souborový slovník a svůj program nebo skript, který zkouší zadávat hesla z daného souboru. Tomuto způsobu prolomení jsem zabránil přidáním tzv. soli k heslu. Solí je myšleno několik nesmyslných znaků, které se připojí k heslu uživatele. Takto spojený řetězec znaků rozhodně nebude v žádném slovníku, a tak útočníkovi znemožní útok tohoto typu.

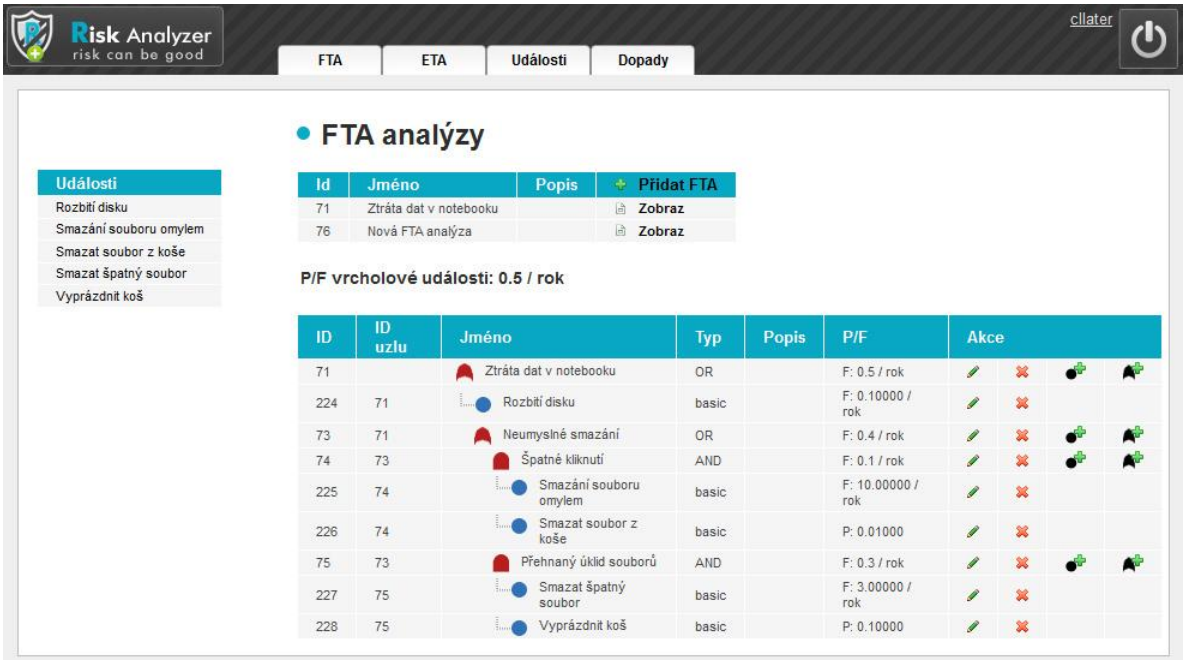
Uživatel se může zaregistrovat v sekci „registrace“ a poté se okamžitě přihlásit v levém menu stránky. Při přihlašování neboli autentizaci uživatele se ověřuje, zda je uživatel opravdu ten, za koho se vydává. PHP nenabízí žádnou standardní implementaci, jak tento proces udělat zabezpečený, a proto se ve většině aplikací objevují bezpečnostní díry. Nette Framework se snaží tyto díry zcela zacelit a zároveň zjednodušit celý proces přihlašování. Pro přihlašování slouží Nette metoda `login`() a pro odhlášení `logout`(). Díky těmto metodám spolu s ověřováním uživatelského jména a hesla jsem implementoval jasnou, přehlednou a bezpečnou autentizaci uživatele.

Po přihlášení uživatele do administrační části se uživateli zobrazí úvodní stránka, na které má možnost změnit své původní heslo za nové. Pokud by dělal uživatel tuto změnu pravidelně, zvýšil by bezpečnost aplikace. Mezi další možnosti nastavení na úvodní obrazovce patří určení časové délky, po kterou se bude provádět analýza (viz kapitola 5.5.1). Z této úvodní obrazovky je možné přejít na jednu z analýz FTA a ETA, nebo si předpřipravit události a dopady pro budoucí analýzy. Zpět na úvodní stranu je možné po kliknutí na logo aplikace umístěné v levé části obrazovky nebo kliknutím na zobrazené přihlašovací jméno uživatele umístěná na opačné straně obrazovky.

6.2 Tvorba FTA

Po načtení stránky pro tvorbu analýzy stromu chyb, jsou zobrazeny doposud uložené analýzy a v levé části obrazovky jsou všechny dostupné události. Při prvním přihlášení jsou všechny položky prázdné. Pro vytvoření první analýzy uživatel musí kliknout na odkaz „Přidat FTA“. Tento odkaz zavolá handler `CreateGate`, který vytvoří v databázi nové hradlo a pomocí tzv. snippetu překreslí tabulku obsahující všechny uložené stromy chyb. Snippet je jedno z pokročilejších maker v Nette Frameworku využívající Ajax, neboli technologie webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačtení. Díky tomuto makru můžeme překreslovat jen část webových stránek označených značkami `{snippet}{/snippet}` a tak zefektivnit práci s aplikací.

Každý strom chyb je možné zobrazit pomocí tlačítka „*Zobraz*“. Při vytvoření nového stromu se vyobrazí tabulka obsahující jeden řádek znázorňující hradlo *OR*. Toto hradlo současně reprezentuje strom chyb, protože jde o hradlo, které nemá žádného předka v tabulce, a tedy se jedná o vrcholovou událost. U každého hradla je v pravé části řádku tabulky možnost editovat dané hradlo, smazat hradlo, přidat novou událost pod hradlo nebo přidat nové hradlo pod toto hradlo. Při přidání hradla nebo události dojde opět k překreslení pouze tabulky pomocí snippetu. Události jsou také brány jako řádek tabulky. Každá událost má opět v pravé části tlačítko pro možnost editace nebo smazaní dané události. Obrazovka s pěti vytvořenými událostmi nalevo a zobrazenou analýzou stromu chyb analyzující frekvenci ztráty dat v notebooku je na obrázku 6.1.



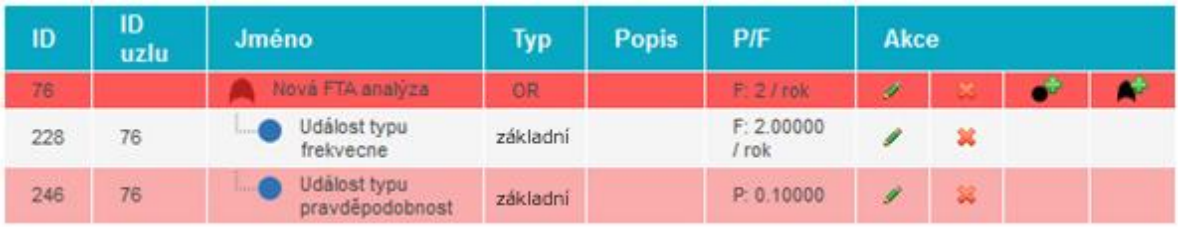
Obrázek 6.1: GUI pro správu FTA

Předem připravené události z levé části obrazovky je možné do stromu chyb vkládat tažením myši. Po kliknutí na událost, se zeleně zobrazí místa, udávající pozici možného vložení dané události do stromu chyb. Ve stromě chyb je možné vkládat události vždy jen pod nějaké hradlo, a tedy budou zeleně podbarvené všechny hradla ve stromě. Přesunem události nad zvolené hradlo, změní hradlo barvu ze zelené na modrou. Tato změna určuje, do jakého hradla jsme se rozhodli událost vložit. Ukázka tohoto procesu je znázorněna na obrázku 6.2. Implementačně je tento proces realizován

pomocí javascriptové knihovny jQuery a její nadstavby jQuery UI. Při výpisu událostí do šablony si do atributu `id` v řádku tabulky vložím `event_xx`, kde `xx` je id události z databáze. Vykreslování hradel do stromu chyb se provádí obdobně. Do atributu `id` hradla vložím `node_xx`, kde `xx` je id hradla z databáze. Při vložení události pod hradlo se z atributu id jak hradla, tak události vyfiltruje pouze `xx`, respektive `id` z databáze daných prvků a s těmito `id` v parametrech se provede metoda realizující uložení události pod hradlo.

Obrázek 6.2: Vkládání událostí do FTA

Jednou z hlavních výhod, oproti většině konkurenčních aplikací je vlastní validace stromu chyb. Pokud se provedou nějaké operace, které nejsou platné pro vkládání událostí a hradel do stromu (viz kapitola 5.5.2.1) automaticky bude zabarvené červeně hradlo a událost které tuto chybu způsobili. Například pokud je pod jedním hradlem *OR* vložena událost typu frekvence a pravděpodobnost, není strom schopen provést výpočet, a tedy podbarví červeně hradlo a událost, která zapříčinila chybu ve výpočtu (viz obrázek 6.3).



Obrázek 6.3: Validace chyb v FTA.

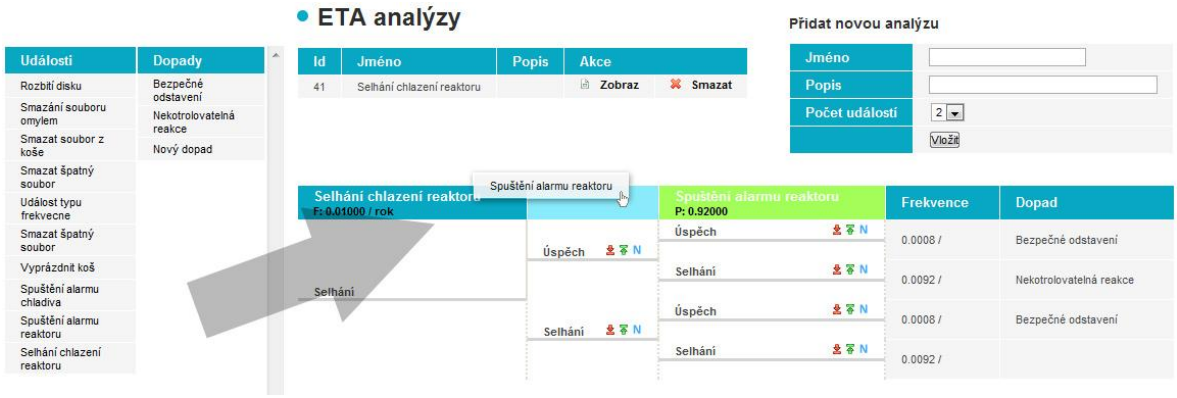
6.3 Tvorba ETA

Obrazovka pro vytváření analýzy stromu událostí je podobná jako pro FTA. Na levé straně obrazovky jsou navíc dopady potřebné pro identifikaci možných výsledků inicializační události. Při

prvním spuštění aplikace jsou i zde všechny tabulky prázdné, a proto je zapotřebí si nejdříve vytvořit několik událostí a dopadů, které se později budou vkládat do stromu událostí. V samotném stromě se události nevytváří, protože ETA v praxi funguje jinak než FTA. U ETA máme pouze jednu inicializační událost, která vede k řadě možných koncových stavů (dopadů). Tuto událost předem známe a známe i frekvenci jejího výskytu. U FTA máme řadu „inicializačních“ událostí, které vedou k jediné vrcholové události. Tyto inicializační události většinou předem neznáme a snažíme se je zjistit. U ETA se směrem od inicializační události v záhlaví stromu vyskytují události, které většinou reprezentují bezpečnostní funkce, o kterých také víme a snažíme se zjistit frekvence možných dopadů. Proto při vytváření nové ETA hned na začátku určujeme počet událostí ve stromě včetně inicializační. Díky této informaci je pak aplikace schopna vykreslit strom a předpřipravit záhlaví stromu pro vkládání událostí tak, aby fungovala přetáhnutím událostí do tabulky pomocí myši podobně jako u FTA. Jednotlivé úrovně stromu v záhlaví mají opět svoje `id`, jak na stránce v atributu `id`, tak v databázi a tažením události nad danou úroveň stromu, se opět zavolá metoda, zajišťující uložení události do úrovně stromu.

Ve stromě událostí opět existují pravidla pro vkládání událostí (viz kapitola 5.5.3.4). Aplikace ale nedovolí uživateli tyto pravidla nijak porušit. Při kliknutí na událost a tažením myši se zvýrazní v záhlaví stromu zeleně ty úrovně, na které je možné danou událost vložit. V atributu `class` u událostí je určeno, o jaký typ události se jedná a tak není možné, aby byla například za inicializační událost vložena událost typu pravděpodobnost, protože první úroveň stromu umožní vložení událostí, které mají atribut `class` jako `freq`. Na obrázku 6.4 je názorná ukázka zabarvení stromu pro událost typu pravděpodobnost.

Pro určování dopadů inicializační události je možné vkládat do stromu předem připravené dopady. Dopady se do stromu vkládají podobně jako události. Implementačně je tento proces realizován také stejně. Při kliknutí na dopad a následné tažení dopadem budou zeleně podbarveny všechny poslední uzly ve stromě. Poslední uzel je totiž vždy spojen s dopadem pro danou větev.



Obrázek 6.4: Vkládání událostí do ETA

U jednotlivých uzlů (kromě inicializačního) je možné nastavit pomocí tří tlačítek, zda se jedná o úspěšný, neúspěšný nebo nezapočítaný uzel. Při tvorbě analýzy se vychází z toho, že inicializační událost je vždy neúspěšná. Po kliknutí na červenou šipku bude uzel brán jako neúspěšný a ponese hodnotu pravděpodobnosti poruchy. Zelená šipka nastavuje uzel na úspěšný a modré tlačítko „N“ nastaví, že při výpočtu výskytu dopadu pro danou větev nebude uzel započítán.

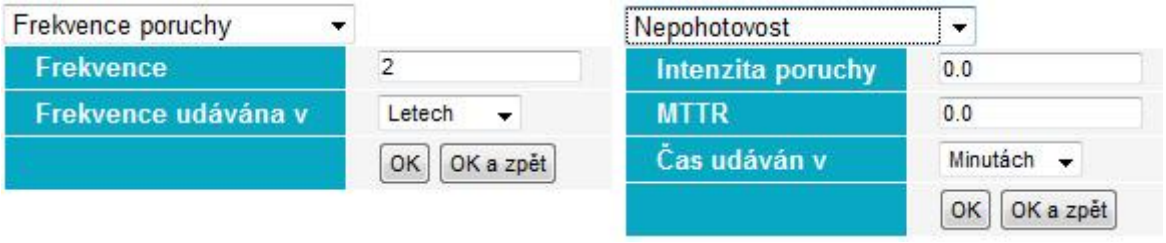
6.4 Tvorba událostí a dopadů

Grafické uživatelské prostředí pro vytváření událostí a dopadů je téměř totožné. V levé horní části obrazovky je tlačítko pro přidávání událostí/dopadů a pod tímto tlačítkem je výpis všech uložených událostí/dopadů a jejich nastavených atributů. Výpis je proveden pomocí tabulky, která je označena jako snippet. Proto při smazání některé z události/dopadu není zapotřebí aktualizovat celé okno ale jen tabulku.

Při vkládání dopadu se pouze vybere název a popis dopadu, ale při vkládání událostí již musí analytik vědět, jaký má vybrat typ vstupních hodnot pro událost. Typy vstupních hodnot jsou popsány v kapitole 5.5.1. Na obrázku 6.5 je grafické rozhraní pro vkládání atributů pro *Událost 1* s konstantní pravděpodobností poruchy 0.4. Pokud by uživatel nenechal zaškrtnuté políčko „*Považovat pravděpodobnost jako konstantní*“, provedl by se výpočet pravděpodobnosti poruchy podle časové délky analýzy, která lze nastavit na úvodní stránce aplikace. Pokud by například byla nastavena délka analýzy na 10 časových jednotek, měla by *Událost 1* v FTA nebo ETA hodnotu pravděpodobnosti podle vzorce $F(t) = 1 - e^{-\Delta t} = 0.98$, což je 98% procentní šance, že událost v tomto časovém rozmezí nastane.

Obrázek 6.5: GUI pro vkládání atributů k události.

Další možné vstupní hodnoty jsou vidět na obrázku 6.6. V levé části obrázku jsou možné vstupní hodnoty pro událost s frekvencí poruchy a v pravé pro nepohotovost události. Nepohotovost události má jako parametr MTTR, a tedy tato událost bude vždy počítána podle časové délky analýzy.



Obrázek 6.6: Parametry při nastavení hodnot událostí s frekvencí a nepohotovostí

7 Případová studie

Tato kapitola slouží pro ověření funkčnosti vytvořeného systému pro analýzy FTA a ETA. Je zřejmé, že o funkčnosti kvalitativní analýzy těchto metod není pochyb, protože tento způsob analýzy slouží pouze ke grafické vizualizaci, ze které muže analytik lépe porozumět mechanismům poruch a časového vývoje nehodové události, ale nic se v této analýze nepočítá. Ověření správnosti kvantitativní analýzy tak lehké není. Proto jsem použil pro ověření jako důvěryhodný vzorek dat u FTA příklad *můstkového obvodu* z normy ČSN EN 61025, popsaného v kapitoly 3.1.5. Pro ověření správnosti systémů ETA jsem použil příklad *selhání chlazení reaktoru* z [2]. ET analýza lze snadno ověřit tím, že součet frekvencí všech dopadů, musí být roven frekvenci inicializační události.

Příklad pro FTA:
Analytik běžně počítá v FTA dvě spolehlivostní charakteristiky. Jsou to očekávaný počet poruch za rok a pravděpodobnost poruchy. V příkladě můstkového obvodu jsem počítal pravděpodobnost poruchy obvodu (viz obrázek 7.1). Pravděpodobnost poruchy vrcholové události vyšla 0.707856, což je stejně jako při výpočtu v kapitole 3.1.5 pomocí Esaryho-Proschanovy metody.

ID	ID uzlu	Jméno	Typ	Popis	P/F
76		Pravděpodobnost poruchy můstkového obvodu	OR		P: 0.7078566151
77	76	AND	AND		P: 0.154
255	77	BLOK A	základní		P: 0.22000
256	77	BLOK B	základní		P: 0.70000
78	76	AND	AND		P: 0.425
257	78	BLOK C	základní		P: 0.85000
258	78	BLOK D	základní		P: 0.50000
79	76	AND	AND		P: 0.066
255	79	BLOK A	základní		P: 0.22000
258	79	BLOK D	základní		P: 0.50000
262	79	BLOK E	základní		P: 0.60000
80	76	AND	AND		P: 0.357
256	80	BLOK B	základní		P: 0.70000
257	80	BLOK C	základní		P: 0.85000
262	80	BLOK E	základní		P: 0.60000

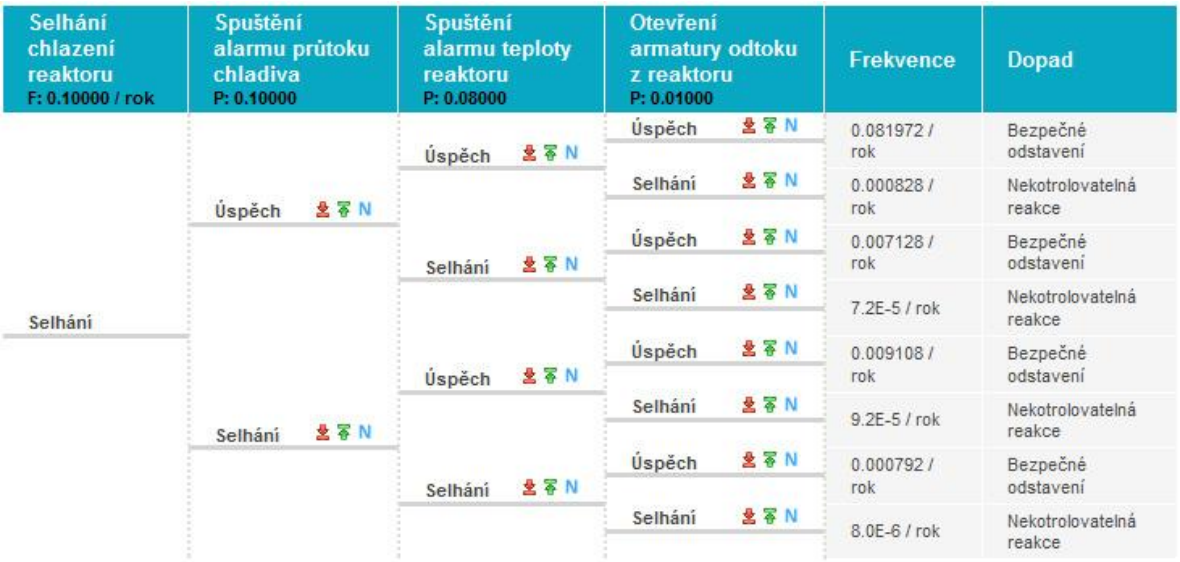
Obrázek 7.1: Výpočet pravděpodobnosti poruchy můstkového obvodu ve vytvořeném systému

Příklad pro ETA:
V mnoha kvantitativních analýzách stromu událostí se počítá s poruchou události spojenou s únikem nebezpečné látky (únik z potrubí, vnitřní exploze atd.). Frekvence takové události se určují z historických záznamů nebo analýzou FTA. Já jsem pro jednoduchost v příkladu *selhání chlazení reaktoru* frekvenci pouze odhadl.

Pokud selže chlazení reaktoru, existují pro tuto událost tři bezpečnostní funkce, mezi které patří:

- Spuštění alarmu průtoku chladiva s pravděpodobností poruchy 0.1
- Spuštění alarmu teploty reaktoru s pravděpodobností poruchy 0.08
- Otevření armatury odtoku z reaktoru s pravděpodobností poruchy 0.01

Obrázek 7.2 ukazuje tento příklad ztráty chladiva v reaktoru s exotermickou reakcí následovanou nekontrolovatelnou reakcí. Z analýzy vyplívají celkem dva možné dopady (bezpečné odstavení a nekontrolovatelná reakce), jež jsou včetně jejich frekvencí zachyceny v obrázku 7.2. Celkový součet frekvencí všech koncových stavů je roven frekvenci inicializační události (0.1 /rok). Tato kontrola je ověřením správnosti konstrukčních a výpočetních vztahů v ETA.



Obrázek 7.2: Výpočet frekvence možných dopadů při selhání chlazení reaktoru

8 Závěr

Cílem této diplomové práce bylo seznámit se s problematikou managementu rizik v projektech IT a prostudovat metody používané v této oblasti. Po prostudování více než patnácti metod jsem se zaměřil na analýzu stromu chyb a analýzu stromu událostí. Pro tyto dvě metody jsem následně navrhl programovou aplikaci, která by měla podpořit kvalitativní a kvantitativní analýzu rizik pomocí těchto dvou metod. Tyto metody se často používají u komplikovanějších systémů, kde dochází k větvení a paralelním procesům, jako například v jaderných elektrárnách, chemických provozech, komunikačních technologiích apod.

Když jsem se touto tématikou postupem času více a více zabýval, začal jsem identifikovat hrozby v mém vlastním podnikání a životě. Začal jsem analyzovat pravděpodobnost vzniku rizika v případě, že nějaká hrozba využije zranitelnost, které mé podnikání a život přináší. Probíraná látka mi začala pomáhat jak v pracovním, tak i osobním životě. Je ale nutné podotknout, že řízení rizik identifikací a analýzou nekončí. Pro každé potenciální riziko je potřeba naplánovat reakci a poté riziko monitorovat a kontrolovat.

Zjistil jsem, že nejdůležitější prvek kvantitativní analýzy je zdroj spolehlivostních dat, od kterého se výsledek analýzy odvíjí. Základním požadavkem na data ke zhodnocení funkčnosti určitého systému jsou počet jednotlivých typů poruch („poruchové módy“) jednotlivých typů zařízení (systém, komponenta) za určitou dobu pozorování a délka trvání těchto poruch (doba opravy) za daných podmínek. Databáze obsahující takovéto zdroje informací jsou k dispozici na konzultační bázi např. pro jaderný, letecký, elektrotechnický a chemický průmysl. V některých zemích a mezinárodních organizacích jsou zřízeny systémy, které obsahují spolehlivostní data. V souvislosti s jaderným průmyslem je ale základna pro statistické zpracování spolehlivostních dat zatím ve vývoji [2].

Při návrhu aplikace jsem se seznámil s několika komplexními aplikacemi zabývajícími se analýzou rizik, které vyvíjejí velké organizace na několika místech světa. Velkou inspirací pro mne byly zkušební verze aplikací *Relex* od společnosti PTC a *FaultTree+* od společnosti Isograph Ltd. Při oslovení se žádostí o zaslání studentské licence mi dokonce společnost Isograph Ltd. okamžitě vyhověla. Od společnosti PTC jsem bohužel dostal odpověď, až když jsem měl aplikaci celou navrženou. Pokud by byl o aplikaci zájem, vidím velký potenciál v jejím rozšíření o další typy vstupních hodnot pro události a nové metody analýz rizik, protože žádný profesionální analytik si nevystačí pouze se dvěma analytickými metodami. V takovémto případě bych na vývoji aplikace nemohl pracovat sám z důvodů časové náročnosti. Také dokonalá znalost všech analytických metod je téměř nadlidský úkol. K tomu aby se aplikace ještě více přiblížila konkurenčním řešením, potřebovala by rozšíření o komponenty např. na vykreslování grafů znázorňujících průběh pravděpodobnosti poruchy, exportování do různěných formátů (Excel, Word, PDF…), přidání více uživatelských pomůcek jako vkládání poznámek, hypertextových odkazů k událostem a stromům atd. Aplikace by mohla být nabízena ve více variantách, čímž by se předešlo nepřehlednému rozhraní přeplněnému funkcemi, jako je tomu u konkurenčních aplikací. Jednou z možností také je, že by aplikace byla vyvíjena pod licencí GNU GPL. Po úspěšném získání části trhu v oblasti aplikací pro analýzu rizik by se nabízela další možná rozšíření aplikace i do jiných oblastí řízení rizik až do celého řízení projektů. Pro takovéto rozšíření by však byl zapotřebí několikaletý výzkum a vývoj.

Hlavním přínosem této práce pro mé studium bylo připomenutí a utřídění základních pojmů, které jsou nezbytnými znalostmi z této oblasti. Zdokonalil jsem se v řízení rizik, o kterém

jsem se dříve dozvěděl pouze v předmětech Management projektů (MPR) a Strategické řízení informačních systémů (SRI) vyučovaných na FIT VUT, kde je látce věnována pouze jedna přednáška. Proto jsem při studiu této oblasti musel využít i mnohé další literatury nejen v českém jazyce. Dále jsem zjistil rozdíly mezi různými metodami pro analýzu rizik a uvědomil si, proč nestačí znát pouze jednu metodu, ale je lepší použít při měření a zkoumání rizik více metod. Čím více jsem o oblasti řízení rizik věděl, tím více jsem si jeho používání začal všímat ve svém okolí. Zjistil jsem, že větší společnosti řízení rizik již delší dobu využívají. Je ovšem otázkou, zda tyto organizace provádějí řízení rizik dostatečně a správně. Také jsem se naučil pracovat v prostředí Nette Framework, které do světa programování v PHP přineslo novou naději a ukázalo, že programování v PHP je stále rychlé, zábavné, moderní a bezpečné. Dále jsem si připomenul výhody používání javascriptové knihovny jQuery, bez které by aplikace nikdy nenabyla tak moderního a efektního dojmu. A v neposlední řadě jsem se naučil používat softwarovou architekturu MVP, která přináší obrovské výhody při případných úpravách a vyvíjení aplikace, což se mi bude v budoucnu jistě hodit stejně jako všechny výše vypsané nové znalosti.

Literatura

[1] ČSN ISO/IEC TR 13335 – 1. *Informační technologie – Směrnice pro řízení bezpečnosti IT – Část 1: Pojetí a modely bezpečnosti IT*. Praha : Český normalizační institut, 1999. 24 s.

[2] PALEČEK,, M., et al. *Postupy a metodiky analýz a hodnocení rizik pro účely zákona o prevenci závažných havárií* [online]. Praha : [s.n.], 2000 [cit. 2011-01-11]. Dostupné z WWW: <www.vubp.cz/index.php/component/docman/doc_download/152-postupy-a-metodiky-analyz-a-hodnoceni-rizik-pro-uely-zakona-o-prevenci-zavanych-havarii>.

[3] Projekt. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 3. 6. 2007, last modified on 5. 1. 2011 [cit. 2011-01-11]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Projekt>.

[4] SCHWALBE, K. *Řízení projektů v IT : Kompletní průvodce*. [s.l.] : Computer Press, a.s., 2007. 720 s. ISBN 80-251-1526-7.

[5] Brainstorming. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21. 1. 2007, last modified on 25. 11. 2010 [cit. 2011-01-11]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Brainstorming>.

[6] ČSN EN 61025. *Analýza stromu poruchových stavů (FTA)*. Praha : ČESKÝ NORMALIZAČNÍ INSTITUT, 2007. 48 s.

[7] MAREK, J. *Analýza rizika s využitím rozhodovacích stromů a analýza jeho citlivosti, včetně simulací. Cideas* [online]. 2005, 1.1.2.1-3, [cit. 2011-01-11]. Dostupný z WWW: <http://www.cideas.cz/free/okno/technicke_listy/2tlv/1121-3.pdf>.

[8] ROUDNÝ, R., LINHART, P. *Krizový management III. – Teorie a praxe rizika*. Pardubice: Univerzita Pardubice, 2007. ISBN 80-7194-924-8.

[9] GARLICK, A. *Estimating Risk : A management Approach*. USA : Gower, 2007. 240 s. ISBN 13-9780566087769.

[10] COOPER, D. F., et al. *Project Risk Management Guidelines*. England : John Wiley & Sons Ltd, 2005. 384 s. ISBN 0-470-02281-7.

[11] BABINEC, F. *Management rizika* [online]. Brno : Slezská Universita v Opavě, 2005. 93 s. Učební text. Slezská Universita v Opavě, Ústav matematiky. Dostupné z WWW: <http://www.math.slu.cz/studmat/AnalyzaRizik/AnalyzaRizik-1.pdf>.

[12] *The Yahoo! User Interface Library (YUI)* [online]. c2009 [cit. 2009-05-01]. Text v angličtině. Dostupný z WWW: <http://developer.yahoo.com/yui/>.

[13] *Nette Framework : Dokumentace* [online]. 2008 [cit. 2011-03-19]. Dostupné z WWW: <http://doc.nette.org/cs/>.

[14] *Dibi : Quick Start* [online]. 2008 [cit. 2011-03-19]. Dostupné z WWW: <http://dibiphp.com/cs/quick-start>.

[15] NetBeans. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21. 12. 2004, last modified on 7. 1. 2011 [cit. 2011-04-19]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/NetBeans>.

[16] *Model View Presenter (MVP) VS Model View Controller (MVC)* [online]. 18.12.2007 [cit. 2011-03-19]. .NET development and architecture. Dostupné z WWW: <http://blog.vuscode.com/malovicn/archive/2007/12/18/model-view-presenter-mvp-vs-model-view-controller-mvc.aspx>.

[17] ZENDULKA, J.; BARTÍK,, V.; KVĚTOŇOVÁ, Š. *Analýza a návrh informačních systémů : Studijní opora*. Brno : Jaroslav Zendulka a kol., 2006. 178 s.

[18] MTTR. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26. 2. 2010, last modified on 19. 3. 2010 [cit. 2011-04-19]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/MTTR>.

[19] Secure Hash Algorithm. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 3. 1. 2008, last modified on 5. 4. 2011 [cit. 2011-04-19]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Secure_Hash_Algorithm>.

Seznam příloh

Příloha 1. CD obsahující:

- složka Risk Analyzer - zdrojové kódy programu
- xjanos00.pdf, xjanos00.docx - dokumentace k diplomové práci
- readme.txt - návod na instalaci aplikace
- databaze.sql - export databáze aplikace
- manual.mp4 – video manuál s rychlým přehled funkcí aplikace