VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VYHODNOCOVÁNÍ SPOLEHLIVOSTNÍCH UKAZATELŮ SYSTÉMŮ ODOLNÝCH PROTI PORUCHÁM

DEPENDABILITY ASSESSMENT OF FAULT TOLERANT SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN SUCHÁNEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JOSEF STRNADEL, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Suchánek Martin**
Program: Informační technologie
Název: **Vyhodnocování spolehlivostních ukazatelů systémů odolných proti poruchám**
Dependability Assessment of Fault Tolerant Systems
Kategorie: Modelování a simulace

Zadání:

1. Proveďte rešerši v oblasti spolehlivosti, zejména v souvislosti s elektronickými systémy. Klasifikujte a charakterizujte i) poruchy a chyby, techniky řízení spolehlivosti se zaměřením na odolnost proti poruchám a ii) prostředky a metody tvorby spolehlivostních modelů a analýzy jejich spolehlivostních ukazatelů.
2. Po dohodě s vedoucím připravte i) seznam systémů a jejich prvků, o jejichž spolehlivostní ukazatele se budete dále zajímat, ii) seznam poruch a jejich vlastností, jejichž vliv na systémy budete dále zkoumat a iii) seznam technik odolnosti proti poruchám, které budete na systémy/prvky aplikovat.
3. Zvolte vhodné prostředky a metody pro tvorbu spolehlivostních modelů a analýzu jejich spolehlivostních ukazatelů.
4. Pro systémy/prvky z bodu 2 vytvořte, s pomocí prostředků a metod z bodu 3, pro různé kombinace poruch a technik odolnosti proti poruchám, jejich spolehlivostní modely a na jejich základě vyhodnoťte spolehlivostní ukazatele těchto systémů/prvků.
5. Na základě výstupu bodu 4 vyhodnoťte schopnost systémů/prvků z bodu 2 odolat, s pomocí zvolených technik, daným poruchám.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Spoľahlivosť je dôležitou súčasťou rôznych systémov. Cieľom práce je vytvorenie spoľahlivostných modelov niektorých opravovaných a neopravovaných systémov odolných proti poruchám a následné vyhodnocovanie ich spoľahlivostných ukazateľov. Na vytváranie modelov je využitý nástroj Uppaal spolu s rozšírením SMC, ktoré slúži na verifikáciu. Výsledkom práce je overenie modelov a vyhodnocovanie spoľahlivostných ukazateľov s využitím nástroja Uppaal SMC.

Abstract

Reliability is an important part of various systems. The aim of the work is to create reliability models of some reconfigurable and nonreconfigurable fault tolerant systems and subsequent evaluation of their reliability indicators. The Uppaal tool is used to create models, along with the SMC extension, which is used for verification. The result of the work is the verification of models and evaluation of reliability indicators using the tool Uppaal SMC.

Kľúčové slová

Spoľahlivosť, porucha, chyba, zlyhanie, odolnosť proti poruchám, Uppaal

Keywords

Dependability, fault, error, failure, fault tolerance, Uppaal

Citácia

SUCHÁNEK, Martin. *Vyhodnocování spolehlivostních ukazatelů systémů odolných proti poruchám.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Vyhodnocování spolehlivostních ukazatelů systémů odolných proti poruchám

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Josefa Strnadela Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Martin Suchánek
12. mája 2021

Poďakovanie

Chcel by som sa poďakovať vedúcemu mojej práce Ing. Josefovi Strnadelovi, Ph.D. za odbornú pomoc a cenné rady, ktoré mi poskytol pri písaní tejto práce a mojej rodine za podporu pri príprave a písaní práce.

Obsah

1 Úvod **3**

2 Spolahlivosť **4**
2.1 Základné pojmy a princípy 4
2.1.1 Atribúty spoľahlivosti 4
2.1.2 Zlyhanie 5
2.1.3 Chyba 5
2.1.4 Porucha 6
2.1.5 Taxonómia porúch 6
2.1.6 Vzťah medzi zlyhaniami, chybami a poruchami 7
2.2 Prostriedky na dosiahnutie spoľahlivosti a bezpečnosti 9
2.2.1 Prevencia porúch 9
2.2.2 Odolnosť proti poruchám 9
2.2.3 Odstránenie poruchy 12
2.2.4 Predpovedanie poruchy 13
2.2.5 Vzťahy medzi prostriedkami spoľahlivosti a bezpečnosti 14
2.3 Vyhodnocovanie spoľahlivostných ukazateľov 16
2.4 Opravované a neopravované systémy 18
2.4.1 Neopravované 18
2.4.2 Opravované 19
2.4.3 Single-point zlyhania 20

3 Prostriedky pre tvorbu spoľahlivostných modelov **21**
3.1 Časované automaty 21
3.2 Nástroj Uppaal 22
3.3 Hlavné časti Uppaalu 22
3.3.1 Systémový editor 23
3.3.2 Simulátor 24
3.3.3 Verifikátor 25
3.4 Rozšírenie Uppaal SMC 26
3.4.1 Syntax v Uppaal SMC 27

4 Modely systémov a experimenty **28**
4.1 Modely v Uppaal SMC 28
4.1.1 Generátor porúch 28
4.1.2 Simplex 29
4.1.3 TMR 29
4.1.4 Degradácia TMR do simplexu 29

4.1.5 TMR so zálohou 30
4.1.6 TMR so single-point zlyhaním 30
4.2 Experimenty 31
4.2.1 Popis experimentov 31
4.2.2 Experiment č.1 32
4.3 Experiment č.2 34
4.4 Experiment č.3 36
4.5 Zhodnotenie 37

5 Záver **39**

Literatúra **40**

A Obsah priloženého CD **41**

Kapitola 1

Úvod

Existujú systémy, pre ktoré sú ich spoľahlivostné ukazatele kľúčové a nachádzajú sa v dôležitých oblastiach ľudského sveta. Je preto dôležité aby sme vedeli tieto ukazatele overiť, odsimulovať a vyhodnotiť. Na tieto činnosti využívame abstraktné a simulačné modely takýchto systémov. Na modeloch môžeme simulovať situácie kedy dôjde k jednej alebo viacerým poruchám, ktoré môžu viesť až k zlyhaniu celého systému. Aby sme takýmto situáciam predišli, využívame prostriedky na zaistenie spoľahlivosti a bezpečnosti.

Táto práca sa zaoberá najskôr zhrnutím dôležitých pojmov z oblasti spoľahlivosti, procesom od výskytu poruchy až po zlyhanie celého systému a prostriedkami na zaistenie spoľahlivosti a bezpečnosti, ako je napríklad odolnosť proti poruchám. Sú tu predstavené niektoré často využívané systémy odolné proti poruchám, ku ktorým sú vytvorené aj ich modely. Tieto modely sú vytvorené v nástroji Uppaal, ktorý tvorí podstatnú časť práce. Je tu zhrnutý jeho význam, jednotlivé časti najmä z uživateľského hľadiska a aj rozširenie SMC, ktoré slúži na verifikáciu modelov. Samostatná časť je potom venované konkrétným modelom opravovaných a neopravovaných systémov odolných proti poruchám. Pri týchto modeloch nás budú zaujímať niektoré spoľahlivostné ukazatele, najmä distribučná funkcia a funkcia hustoty pravdepodobnosti.

Na záver bude zhodnotenie dosiahnutých výsledkov a zamyslenie sa nad ďalšou prípadnou prácou a návrhmi.

Kapitola 2

Spolahlivosť

Spoľahlivosť je schopnosť poskytovať služby, ktorým možno oprávnene dôverovať. V súvislosti s elektronickými systémami je definovaná ako spoľahlivosť systému. Je to schopnosť systému vyhnúť sa častejším a závažnejším poruchám, ako je prijateľné.[2]

Pôvodná definícia je všeobecnou definíciou, ktorej cieľom je zovšeobecniť klasickejšie pojmy dostupnosť, spoľahlivosť, bezpečnosť, integrita, udržiavateľnosť atď., ktoré sa neskôr stanú atribútmi spoľahlivosti. Alternatívna definícia spoľahlivosti vychádza z argumentu, že systém môže zlyhať a zvyčajne aj zlyhá. Zaoberá sa otázkami. Je to však stále spoľahlivé? Kedy sa stane nespoľahlivým? Alternatívna definícia teda poskytuje kritérium pre rozhodnutie, či je alebo nie je možné napriek výpadkom služieb považovať systém za spoľahlivý. Pojem zlyhanie spoľahlivosti, ktorý je priamo odvodený z tejto definície, navyše umožňuje nadviazať spojenie s vývojovými zlyhaniami.

2.1 Základné pojmy a princípy

2.1.1 Atribúty spoľahlivosti

Atribúty spoľahlivosti môžu mať rôzny význam v závislosti od aplikácie určenej pre daný výpočtový systém. Vo všeobecnosti sa vyžadujú dostupnosť, integrita a udržiavateľnosť, aj keď v rôznej miere v závislosti od aplikácie, zatiaľ čo spoľahlivosť, bezpečnosť a dôvernosť, môžu alebo nemusia byť vyžadované v závislosti od aplikácie. Z dôvodu nevyhnutnej prítomnosti alebo výskytu porúch, nie sú systémy nikdy úplne dostupné, spoľahlivé, bezpečné alebo chránené.

- *Dostupnosť* - pripravenosť pre správnu službu
- *Spoľahlivosť* – kontinuita správnej služby
- *Bezpečnosť* - absencia katastrofických následkov na uživateľov a životné prostredie
- *Integrita* – absencia nesprávnych zmien systému
- *Udržiavateľnosť* – schopnosť podstúpiť úpravy a opravy
- *Dôvernosť* – absencia neoprávneného zverejnenia informácií

Okrem vyššie spomínaných atribútov je možné definovať aj ďalšie, sekundárne, atribúty, ktoré spresňujú alebo špecializujú primárne atribúty. Príklady takýchto sekundárnych atribútov sú:

- *Robustnosť* - spoľahlivosť vzhľadom na vonkajšie poruchy, ktorá charakterizuje reakciu systému na konkrétnu triedu porúch
- *Zodpovednosť* – dostupnosť a integrita totožnosti osoby, ktorá vykonala operáciu
- *Autenticita* - integrita obsahu a pôvodu správy a prípadne ďalších informácií, napríklad času vysielania
- *Neodmietnuteľnosť* – dostupnosť a integrita identity odosielateľa správy (neodmietnutie pôvodu) alebo príjemcu (neodmietnutie prijatia).

Obr. 2.1: Atribúty spoľahlivosti.

2.1.2 Zlyhanie

Zlyhanie služby je udalosť, ktorá nastane v prípade, že sa doručená služba odchýli od správnej služby. Služba zlyhá buď preto, že nie je v súlade s funkčnou špecifikáciou, alebo preto, že táto špecifikácia nedostatočne opísala funkciu systému. Zlyhanie služby je prechod zo správnej služby na nesprávnu službu, t. j. na neimplementovanú funkciu systému. Obdobie dodania nesprávnej služby je výpadok služby. Prechod od nesprávnej služby k správnej službe je obnovenie služby. Odchýlka od správnej služby môže mať rôzne formy, ktoré sa nazývajú režimy zlyhania služby a sú zoradené podľa závažnosti zlyhania.

Pretože služba je postupnosťou externých stavov systému, zlyhanie služby znamená, že najmenej jeden (alebo viac) externých stavov systému sa líši od správneho stavu. Odchýlka sa nazýva chyba. Usudzovaná alebo predpokladaná príčina chyby sa nazýva porucha.

Keď funkčná špecifikácia systému obsahuje množinu niekoľkých funkcií, zlyhanie jednej alebo viacerých služieb implementujúcich tieto funkcie môže nechať systém v zhoršenom režime, ktorý užívateľovi stále ponúka podmnožinu potrebných služieb. Špecifikácia môže identifikovať niekoľko takýchto režimov, napríklad pomalú službu, obmedzenú službu, pohotovostnú službu atď. Tu hovoríme, že systém utrpel čiastočné zlyhanie svojej funkčnosti alebo výkonu.

2.1.3 Chyba

Chyba je časť celkového stavu systému, ktorá môže viesť k zlyhaniu. Zlyhanie nastane, keď chyba spôsobí odchýlku doručenej služby od správnej. Príčina chyby sa nazýva poru-

cha. Chyba sa zistí, ak je jej prítomnosť indikovaná chybovým hlásením alebo chybovým signálom. Chyby, ktoré sú prítomné, ale nie sú zistené, sa nazývajú skryté chyby.

Pretože systém pozostáva z množiny navzájom sa ovplyvňujúcich komponentov, celkový stav je množinou stavov jeho komponentov. Z definície vyplýva, že porucha spôsobí najskôr chybu v stave jedného (alebo viacerých) komponentov, ale k zlyhaniu služby nedôjde, pokiaľ externý stav tohto komponentu nie je súčasťou externého stavu systému. Kedykoľvek sa chyba stane súčasťou externého stavu komponentu, dôjde k zlyhaniu služby tohto komponentu, ale chyba zostáva interná v celom systéme. To, či chyba skutočne povedie k zlyhaniu služby, závisí od dvoch faktorov:

1. Štruktúra systému, a najmä povaha akejkoľvek redundancie, ktorá v ňom existuje:
   - ochranná redundancia zavedená s cieľom zabezpečiť odolnosť proti poruchám, ktorá je výslovne určená na zabránenie tomu, aby chyba viedla k zlyhaniu služby,
   - neúmyselná redundancia (v praxi je ťažké, ak nie je nemožné vytvoriť systém bez akejkoľvek formy redundancie), ktorá môže mať rovnaký, pravdepodobne neočakávaný, výsledok ako úmyselná redundancia.
2. správanie systému - časť stavu, ktorá obsahuje chybu, nemusí byť pre službu nikdy potrebná alebo môže byť chyba odstránená (napr. prepísaná) skôr, ako povedie k zlyhaniu.

Niektoré poruchy (napr. výbuch elektromagnetického žiarenia) môžu súčasne spôsobiť chyby vo viac ako jednej súčasti. Takéto chyby sa nazývajú viacnásobné súvisiace chyby. Samostatné chyby sú chyby, ktoré ovplyvňujú iba jeden komponent.

2.1.4 Porucha

Poruchy môžu byť interné alebo externé. Predchádzajúca prítomnosť vnútornej poruchy, ktorá umožňuje externej poruche poškodiť systém, je nevyhnutná na to, aby externá porucha spôsobila chybu a prípadne následné zlyhanie. Vo väčšine prípadov porucha najskôr spôsobí chybu v servisnom stave komponentu, ktorý je súčasťou interného stavu systému a externý stav nie je okamžite ovplyvnený. Z tohto dôvodu je definícia chyby časťou celkového stavu systému, ktorá môže viesť k následnému zlyhaniu služby. Je dôležité poznamenať, že veľa chýb nedosahuje externý stav systému a spôsobuje zlyhanie. Porucha je aktívna, keď spôsobí chybu, inak je neaktívna.

2.1.5 Taxonómia porúch

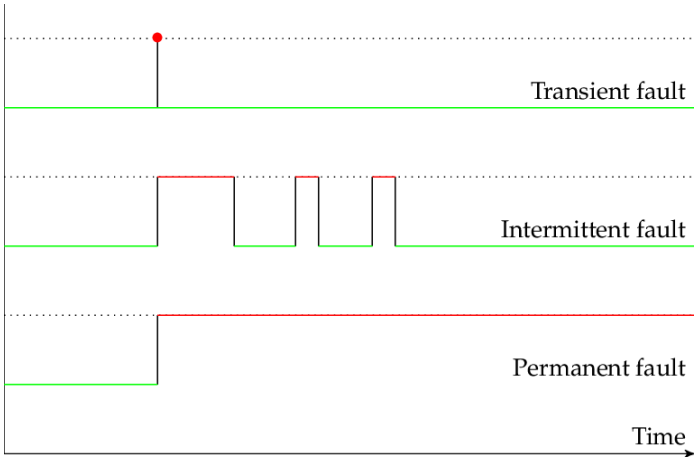
Všetky poruchy, ktoré môžu ovplyvniť systém počas jeho životnosti, sú klasifikované podľa ôsmich základných hľadísk, ktoré vedú k elementárnym triedam porúch. Keby boli možné všetky kombinácie ôsmich základných tried porúch, bolo by 256 rôznych kombinácií poruchových tried. Nie všetky kritériá však možno uplatniť na všetky triedy porúch. Je známych 31 pravdepodobných kombinácií. V budúcnosti bude možné identifikovať ešte viac takýchto kombinácií. Tieto kombinované triedy porúch možno rozdeliť do troch hlavných, čiastočne sa prekrývajúcich zoskupení:

- *poruchy vývoja*, ktoré zahŕňajú všetky triedy porúch vyskytujúce sa počas vývoja,
- *fyzické poruchy*, ktoré zahŕňajú všetky triedy porúch ovplyvňujúce hardvér,

- *poruchy interakcie*, ktoré zahŕňajú všetky vonkajšie poruchy.

Podľa časovej náročnosti môžeme poruchy zatriediť do nasledujúcich troch kategórií, ako je znázornené na obrázku 2.2:

- *prechodná porucha* - (angl. transient fault), má často konečnú dobu trvania kratšiu ako stabilný časový interval ovplyvneného signálu, vyskytne sa iba raz a potom zmizne. Takáto chyba môže byť dôsledkom vonkajších príčin, ako je rušenie. Prechodné poruchy sú spravidla najbežnejšie a je náročné ich identifikovať, pretože môžu po chybách zmiznúť.[6]
- *občasná porucha* - (angl. intermittent fault), táto porucha opakovane mizne, systém týmto pádom opakovane prechádza medzi normálnym a poruchovým stavom. Môže to byť spôsobené nestabilnou činnosťou zariadenia.[6]
- *trvalá porucha* - (angl. permanent fault), táto porucha nezmizne, systém ostáva v poruchovom stave, pokiaľ sa v systéme nevykonajú opravné opatrenia. Trvalá chyba je zvyčajne spôsobená poruchami subsystému, fyzickým poškodením alebo chybou návrhu.[6]



Obr. 2.2: Poruchy podľa časovej náročnosti[6].

2.1.6 Vzťah medzi zlyhaniami, chybami a poruchami

Mechanizmy vytvárania a prejavovania porúch, chýb a zlyhaní sú zhrnuté nasledovne:

1. Porucha je aktívna, ak spôsobí chybu, inak je neaktívna. Aktívna porucha je buď:
   - *interná porucha*, ktorá bola predtým neaktívna a bola aktivovaná výpočtovým procesom alebo podmienkami prostredia,
   - *externá porucha*, aktivácia poruchy je aplikácia vstupu (aktivačný vzor) na komponent, ktorý spôsobí aktiváciu nečinnej poruchy. Väčšina vnútorných porúch sa cyklí medzi ich nečinným a aktívnym stavom.

2. Šírenie chýb v rámci daného komponentu (tzv. vnútorné šírenie) je spôsobené výpočtovým procesom: Chyba sa postupne transformuje na ďalšie chyby. Šírenie chýb z komponentu A do komponentu B, ktoré prijíma službu z A (tj. externé šírenie), nastane, keď sa interným šírením chyba dostane do servisného rozhrania komponentu A. V tomto okamihu sa služba dodávaná z A do B stane nesprávnou a následné zlyhanie služby A sa javí ako externá porucha B a šíri chybu do B prostredníctvom svojho užívateľského rozhrania.

3. Zlyhanie služby nastane, keď sa chyba rozšíri do servisného rozhrania a spôsobí, že sa služba dodávaná systémom líši od správnej služby. Zlyhanie komponentu spôsobí trvalú alebo prechodnú poruchu v systéme, ktorý obsahuje komponent. Zlyhanie služby systému spôsobí trvalú alebo prechodnú externú poruchu pre ostatné systémy, ktoré prijímajú službu z daného systému.

Tieto mechanizmy umožňujú dokončenie „reťazca hrozieb“. Šípky v tomto reťazci vyjadrujú príčinnú súvislosť medzi zlyhaniami, chybami a poruchami. Mali by sa interpretovať všeobecne, šírením sa dá vygenerovať niekoľko chýb skôr, ako dôjde k zlyhaniu. Je potrebné zdôrazniť, že z vyššie uvedených mechanizmov môže dôjsť k šíreniu, a teda inštanciám tohto reťazca prostredníctvom interakcie medzi komponentmi alebo systémami, zložením komponentov do systému a vytvorením alebo úpravou systému.

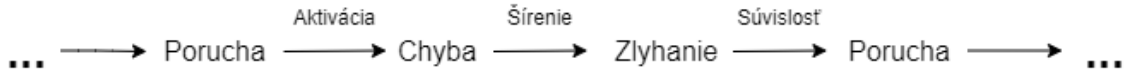
Schopnosť identifikovať vzor aktivácie poruchy, ktorá spôsobila jednu alebo viac chýb, je reprodukovateľnosť aktivácie poruchy. Poruchy je možné kategorizovať podľa ich aktivovateľnej reprodukovateľnosti: Poruchy, ktorých aktivácia je reprodukovateľná, sa nazývajú pevné alebo tvrdé, zatiaľ čo poruchy, ktorých aktivácia nie je systematicky reprodukovateľná, sú nepolapiteľné alebo jemné. Väčšina zostávajúcich vývojových porúch vo veľkom a zložitom softvéri sú nepolapiteľné poruchy.Sú dostatočne zložité na to, aby ich aktivačné podmienky záviseli od zložitých kombinácií vnútorného stavu a vonkajších požiadaviek, ktoré sa vyskytujú zriedka a je veľmi ťažké ich reprodukovať. Ďalšie príklady nepolapiteľných porúch sú:

- „vzorovo citlivé“ poruchy v polovodičových pamätiach, zmeny parametrov hardvérového komponentu (účinky teplotných výkyvov, oneskorenie časovania v dôsledku parazitnej kapacity atď.),
- podmienkami ovplyvňujúce hardvér alebo softvér, ktoré nastanú, keď zaťaženie systému prekročí určitú úroveň, čo spôsobí napríklad okrajové načasovanie a synchronizáciu.

Často sa vyskytujú situácie týkajúce sa viacerých porúch alebo zlyhaní. Zlyhania systému sa často ukážu pri neskoršom preskúmaní, že boli spôsobené chybami, ktoré sú zapríčinené radom rôznych súčasne sa vyskytujúcich porúch. V prípade systému s definovanými hranicami je jedinou poruchou tá, ktorá je spôsobená jednou nepriaznivou fyzickou udalosťou alebo jedným škodlivým ľudským činom. Viaceré poruchy sa skladajú z dvoch alebo viacerých súbežných, prekrývajúcich sa alebo postupných jednotlivých porúch, ktorých následky, t. j. chyby, sa časovo prekrývajú. To znamená, že chyby spôsobené týmito poruchami sú v systéme prítomné súčasne. Zváženie viacerých porúch vedie k rozlíšeniu:

- *nezávislých porúch*, ktoré sa pripisujú rôznym príčinám,
- *súvisiacich porúch*, ktoré sa pripisujú spoločnej príčine.

Súvisiace poruchy všeobecne spôsobujú podobné chyby, t. j. chyby, ktoré sa nedajú rozlíšiť použitými detekčnými mechanizmami, zatiaľ čo nezávislé poruchy zvyčajne spôsobujú odlišné chyby. Môže sa však stať, že nezávislé poruchy vedú k podobným chybám alebo že súvisiace poruchy vedú k odlišným chybám. Zlyhania spôsobené podobnými chybami sú zlyhania v spoločnom režime.



Obr. 2.3: Základný reťazec hrozieb spoľahlivosti a bezpečnosti.

2.2 Prostriedky na dosiahnutie spoľahlivosti a bezpečnosti

V priebehu posledných rokov bolo vyvinutých množstvo prostriedkov na dosiahnutie rôznych atribútov spoľahlivosti a bezpečnosti. Tieto prostriedky možno rozdeliť do štyroch hlavných kategórií:

- *prevencia porúch* - predchádzanie vzniku alebo zavádzaniu porúch
- *odolnosť proti poruchám* – predchádzanie poruchám služby v prípade výskytu chýb
- *odstránenie poruchy* - zníženie počtu a závažnosti porúch
- *predpovedanie porúch* – odhad súčasného počtu, budúcej incidencie a pravdepodobných následkov porúch.

Cieľom prevencie a odolnosti proti poruchám je poskytnúť schopnosť dodať službu, ktorej sa dá dôverovať, zatiaľ čo odstraňovanie porúch a predpovedanie porúch sa zameriavajú na dosiahnutie dôvery v túto schopnosť odôvodnením, že funkčné špecifikácie, špecifikácie spoľahlivosti a bezpečnosti sú primerané, a že systém ich pravdepodobne využije.

2.2.1 Prevencia porúch

Prevencia porúch je súčasťou všeobecného inžinierstva. Niektoré aspekty prevencie porúch sú priamo v záujme spoľahlivosti a bezpečnosti a je možné o nich diskutovať v závislosti na triedy porúch. Prevencia vývojových porúch je hlavným cieľom vývojových metodík, a to tak pre softvér (napr. skrývanie informácií, modularizáciu, používanie programovacích jazykov so silným typom), ako aj pre hardvér (napr. pravidlá návrhu). Vylepšenie vývojových procesov s cieľom znížiť počet porúch zavedených do vyrábaných systémov je založené na zaznamenávaní porúch vo výrobkoch a odstraňovaní príčin porúch pomocou úprav procesu.

2.2.2 Odolnosť proti poruchám

Techniky odolnosti proti poruchám

Odolnosť proti poruchám je zameraná na predchádzanie zlyhaniam. Vykonáva sa prostredníctvom detekcie chýb a obnovenia systému. Zvyčajne po spracovaní poruchy nasleduje nápravná údržba zameraná na odstránenie porúch, ktoré boli izolované manipuláciou s

nimi. Inak povedané, faktorom, ktorý odlišuje odolnosť proti poruchám od údržby, je to, že údržba si vyžaduje účasť externého agenta. Uzavreté systémy sú tie systémy, pri ktorých odstránenie poruchy nie je možné prakticky implementovať (napr. hardvér sondy hlbokého vesmíru).[7]

Rollback a rollforward sa vyvolávajú na požiadanie po vykonaní detekcie chyby, zatiaľ čo kompenzáciu je možné uplatniť buď na požiadanie, alebo systematicky, vo vopred určených časoch alebo udalostiach, nezávisle od prítomnosti alebo neprítomnosti (zistenej) chyby. Spracovanie chýb na požiadanie a následné spracovanie porúch spolu s obnovou systému, odtiaľ názov zodpovedajúcej stratégie odolnosti proti poruchám. Detekcia chýb a obnova systému alebo jednoducho detekcia a obnova.

Maskovanie porúch alebo jednoducho maskovanie vyplýva zo systematického využívania kompenzácie. Takéto maskovanie zakryje možnú progresívnu a nakoniec smrteľnú stratu ochrannej redundancie. Praktické implementácie maskovania teda zvyčajne zahŕňajú detekciu chýb (a pravdepodobne aj spracovanie porúch), čo vedie k maskovaniu a zotaveniu. Je zaujímavé, že:

1. Rollback a rollforward sa navzájom nevylučujú. Je možné sa najskôr pokúsiť o rollback, ak chyba pretrváva, je možné pokúsiť sa o rollforward.

2. Občasné poruchy nevyžadujú izoláciu alebo rekonfiguráciu, identifikáciu toho, či je porucha prerušovaná alebo nie, je možné vykonať buď spracovaním chyby (opakovanie chyby naznačuje, že porucha nie je prerušovaná), alebo diagnostikou poruchy, keď sa použije funkcia rollforward.

3. Spracovanie porúch môže priamo nasledovať po detekcii chýb bez toho, aby ste sa pokúsili o ich spracovanie.

Preventívne zisťovanie a manipulácia s chybami, po ktorých môže prípadne nasledovať manipulácia s poruchami, sa bežne vykonáva pri zapnutí systému. Taktiež prichádza do úvahy počas prevádzky v rôznych formách, ako je kontrola rezervy, čistenie pamäte, programy auditu alebo takzvané omladenie softvéru, zamerané na odstránenie účinkov starnutia softvéru skôr, ako dôjde k jeho zlyhaniu.

Obr. 2.4: Pokrytie odolnosti proti poruchám.

Implementácia odolnosti proti poruchám

Výber techník detekcie chýb, manipulácie s chybami a manipulácie s poruchami a ich implementácie priamo súvisí a je veľmi závislý od základného predpokladu poruchy. Triedy porúch, ktoré je možné skutočne tolerovať, závisia od predpokladu poruchy.To sa zvažuje v procese vývoja, a teda sa spolieha na nezávislosť prepúšťania, pokiaľ ide o proces vytvárania a aktivácie porúch. Široko používanou metódou dosiahnutia odolnosti proti poruchám je vykonávanie viacerých výpočtov prostredníctvom viacerých kanálov, sekvenčne alebo súbežne. Ak sa predpokladá odolnosť proti fyzickým poruchám, kanály môžu mať identickú konštrukciu založenú na predpoklade, že hardvérové komponenty zlyhajú nezávisle. Takýto prístup sa ukázal ako adekvátny pre nepolapiteľné vývojové poruchy, a to prostredníctvom rollbacku, nie je to však vhodné pre odolnosť proti solídnym vývojovým poruchám, čo si vyžaduje, aby kanály implementovali rovnakú funkciu prostredníctvom samostatných návrhov a implementácií, t. j. prostredníctvom rozmanitosti návrhov. Poskytnutie požadovanej funkčnej schopnosti spracovania v súčasti spolu s mechanizmami súbežnej detekcie chýb vedie k poňatiu samokontrolnej súčasti, či už v hardvéri alebo v softvéri.

Je zrejmé, že nie všetky techniky odolnosti proti poruchám sú rovnako účinné. Miera účinnosti ktorejkoľvek danej techniky odolnosti proti poruchám sa nazýva jej pokrytie. Nedokonalosti odolnosti proti poruchám, t. j. nedostatočné pokrytie odolnosti proti poruchám, predstavujú závažné obmedzenie zvyšovania spoľahlivosti, ktoré je možné dosiahnuť. Takéto nedokonalosti odolnosti proti poruchám sú buď dôsledkom:

1. vývojovej chyby ovplyvňujúcej mechanizmy odolnosti proti poruchám s ohľadom na predpoklady chýb uvedených počas vývoja, ktorých dôsledkom je nedostatok pokrytia chýb a ich riešenia (definované vzhľadom na triedu chýb alebo porúch, napr. jednotlivé chyby, zaseknuté poruchy atď. ako podmienená pravdepodobnosť, že technika je účinná vzhľadom na to, že sa chyby alebo poruchy vyskytli),

2. predpokladaných porúch, ktoré sa líšia od porúch skutočne sa vyskytujúcich v prevádzke, ktoré vedú k nedostatku pokrytia predpokladov porúch, ktoré môžu byť následne spôsobené buď

   - *zlyhaním komponentov*, ktoré sa nesprávajú tak, ako sa predpokladalo, to je nedostatkom režimu pokrytia porúch,
   - *výskytom spoločného režimu porúch*, v prípade nezávislých, je to nedostatok nezávislosti pokrytia zlyhania.

Nedostatok pokrytia chybami a poruchami sa ukázal ako drastická limit pre zlepšenie spoľahlivosti. Podobné efekty môžu vyplynúť z nedostatku pokrytia zlyhania, konzervatívne predpoklady porúch (napr. byzantské chyby) budú mať za následok vyššie pokrytie zlyhania, a to na úkor nevyhnutnosti zvýšenia redundancie a zložitejších mechanizmov odolnosti proti poruchám, ktoré môžu viesť k celkovému zníženiu spoľahlivosti a bezpečnosti systému.

Dôležitým problémom v koordinácii činností viacerých komponentov je zabránenie šíreniu chýb, ktoré neovplyvní činnosť nefunkčných komponentov. Tento problém sa stáva obzvlášť dôležitým, keď daný komponent potrebuje komunikovať niektoré informácie s ostatnými komponentmi. Typickými príkladmi takýchto informácií o jednom zdroji sú údaje miestnych senzorov, hodnota miestnych hodín, lokálne zobrazenie stavu ostatných komponentov atď. Dôsledkom tejto potreby prenosu informácií z jedného zdroja, z jedného komponentu do iných komponentov je, že komponenty, ktoré nezlyhali musia dosiahnuť dohodu

o tom, ako by mali vzájomne a konzistentne využívať informácie, ktoré získavajú. Tento problém sa nazýva problém konsenzu. Systematické zavádzanie odolnosti proti poruchám často uľahčuje pridanie podporných systémov špecializovaných na odolnosť proti poruchám (napr. softvérové monitory, servisné procesory, vyhradené komunikačné spojenia).

Reflexia, technika transparentného a vhodného rozšírenia všetkých relevantných akcií objektu alebo softvérového komponentu, napr. s cieľom zabezpečiť, že tieto akcie je možné v prípade potreby vrátiť späť, sa dá použiť v objektovo orientovanom softvéri a prostredníctvom poskytovania middlewaru.

2.2.3 Odstránenie poruchy

Odstraňovanie poruchy môže nastať v procese vývoja systému, alebo v čase jeho používania.

Odstránenie poruchy počas vývoja

Odstránenie poruchy počas fázy vývoja životného cyklu systému pozostáva z troch krokov: overenie, diagnostika a oprava. Po overení nasleduje proces kontroly, či systém dodržiava dané vlastnosti, ktoré sa nazývajú overovacie podmienky. Ak nie, je potrebné podniknúť ďalšie dva kroky a to diagnostikovať poruchu (poruchy), ktoré bránili splneniu podmienok overenia, a potom vykonať potrebné opravy. Po oprave by sa mal proces overovania zopakovať, aby sa skontrolovalo, či odstránenie poruchy nemalo nežiaduce následky. Overenie vykonané v tejto fáze sa zvyčajne nazýva regresné overenie. Kontrola špecifikácie sa zvyčajne nazýva validácia. K odhaleniu porúch špecifikácie môže dôjsť v ktorejkoľvek fáze vývoja, či už počas samotnej fázy špecifikácie, alebo počas ďalších fáz, keď sa zistí dôkaz, že systém nebude implementovať svoju funkciu, alebo že implementáciu nie je možné dosiahnuť nákladovo efektívnym spôsobom.

Techniky overovania je možné klasifikovať podľa toho, či zahŕňajú vykonávanie systému alebo nie. Overenie systému bez skutočného vykonania je statické overenie. Takéto overenie je možné vykonať:

1. na *samotnom systéme*, vo forme:
    - *statickej analýzy*, napr. inšpekcie alebo prehliadky, analýza toku údajov, analýza zložitosti, abstraktná interpretácia, kontroly kompilátorov, hľadanie zraniteľnosti atď.
    - *dokazovania teorému;*
2. na *modeli správania systému*, kde sú obvykle modelom prechody stavov (Petriho siete, konečné automaty), čo vedie ku kontrole modelu.

Overenie systému jeho spúšťaním predstavuje dynamické overenie, vstupy dodávané do systému môžu byť buď symbolické v prípade symbolického vykonania, alebo skutočné v prípade overovacieho testovania, zvyčajne sa jednoducho nazýva testovanie. Dôkladné testovanie systému so zreteľom na všetky jeho možné vstupy je všeobecne nepraktické. Metódy na stanovenie testovacích vzorov možno klasifikovať podľa dvoch hľadísk, kritérií pre výber vstupov pre test a generovania vstupov pre test.

Pozorovanie výstupov testu a rozhodnutie, či spĺňajú alebo nespĺňajú overovacie podmienky, sa nazýva oracle problém. Podmienky overenia sa môžu vzťahovať na celú sadu výstupov alebo na ich kompaktnú funkciu (napr. podpis systému pri testovaní na fyzické poruchy v hardvéri alebo na „čiastočné oracle“ pri testovaní na poruchy vývoja softvéru).

Pri testovaní na fyzické poruchy sa výsledky, kompaktné alebo nie, očakávané od testovaného systému pre danú vstupnú sekvenciu, určujú simuláciou alebo referenčným systémom (zlatá jednotka). Pri poruchách vývoja je referenciou obvykle špecifikácia, môže to byť tiež prototyp alebo iná implementácia rovnakej špecifikácie v prípade dizajnovej rozmanitosti (testovanie typu back-to-back).

Metódy overovania sa môžu použiť v kombinácii. Napríklad na uľahčenie určenia testovacích vzorov je možné použiť symbolické prevedenie, na overenie vlastností modelov nekonečného stavu možno použiť dôkaz o vete a na porovnanie rôznych testovacích stratégií možno použiť mutačné testovanie. Pretože overovanie je potrebné vykonať počas celého vývoja systému, vyššie uvedené techniky sú použiteľné pre rôzne formy, ktoré systém má počas svojho vývoja ako sú prototyp, komponent atď. Vyššie uvedené techniky sa uplatňujú aj pri overovaní mechanizmov odolnosti proti poruchám, najmä:

- formálne statické overovanie,
- testovanie, pri ktorom je potrebné, aby poruchy alebo chyby boli súčasťou testovacích vzorcov, čo sa zvyčajne označuje ako injektáž porúch.

Overenie, či systém nedokáže viac, ako je stanovené, je obzvlášť dôležité z hľadiska toho, čo by systém nemal robiť, teda z hľadiska bezpečnosti a ochrany.

Odstránenie poruchy počas používania

Odstránenie poruchy počas používania systému je nápravná alebo preventívna údržba. Cieľom opravnej údržby je odstránenie porúch, ktoré spôsobili jednu alebo viac chýb a ktoré boli hlásené, zatiaľ čo preventívna údržba je zameraná na odhalenie a odstránenie porúch skôr, ako by mohli spôsobiť chyby počas normálnej prevádzky. Neskoré poruchy zahŕňajú:

- *fyzické poruchy*, ktoré sa vyskytli od posledných preventívnych opatrení údržby,
- *poruchy vývoja*, ktoré viedli k chybám v iných podobných systémoch.

Nápravná údržba vývojových porúch sa zvyčajne vykonáva postupne. Poruchu je možné najskôr izolovať (napr. riešením alebo opravou) pred dokončením samotného odstránenia. Tieto formy údržby sa uplatňujú na systémy, ktoré nie sú odolné proti poruchám, aj na systémy odolné proti poruchám, ktoré je možné udržiavať online (bez prerušenia poskytovania služby) alebo offline (počas výpadku služby).

2.2.4 Predpovedanie poruchy

Predpovedanie porúch sa vykonáva vykonaním vyhodnotenia správania systému z hľadiska výskytu alebo aktivácie poruchy. Hodnotenie má dva aspekty:

- *kvalitatívne* alebo *poradové hodnotenie*, ktorého cieľom je identifikovať, ohodnotiť a klasifikovať režimy zlyhania alebo kombinácie udalostí (poruchy komponentov alebo podmienky prostredia), ktoré by viedli k zlyhaniu systému,
- *kvantitatívne* alebo *pravdepodobnostné hodnotenie*, ktorého cieľom je vyhodnotiť z hľadiska pravdepodobnosti mieru, do akej sú uspokojené niektoré z atribútov, tieto atribúty sa potom považujú za miery.

Metódy kvalitatívneho a kvantitatívneho hodnotenia sú buď špecifické (napr. analýza zlyhania a analýza účinkov pre kvalitatívne hodnotenie alebo Markovove reťazce a stochastické Petriho siete pre kvantitatívne hodnotenie), alebo ich možno použiť na vykonanie obidvoch foriem hodnotenia (napr. blok spoľahlivosti, poruchové stromy).

Dva hlavné prístupy k pravdepodobnostnému predpovedaniu porúch, ktorých cieľom je odvodiť pravdepodobnostné odhady, sú modelovanie a (hodnotiace) testovanie. Tieto prístupy sa dopĺňajú, pretože pri modelovaní sú potrebné údaje o modelovaných základných procesoch (proces zlyhania, proces údržby, proces aktivácie systému atď.), ktoré je možné získať testovaním alebo spracovaním údajov o zlyhaniach.

Modelovanie je možné vykonať s ohľadom na fyzické poruchy, vývojové poruchyy alebo kombináciu oboch. Skladá sa z dvoch fáz:

- konštrukcia modelu systému z elementárnych stochastických procesov, ktoré modelujú správanie komponentov systému a ich interakcie. Tieto základné stochastické procesy súvisia so zlyhaniami, s obnovou služby vrátane opráv a s pracovným cyklom systému alebo fázami činnosti,
- spracovanie modelu na získanie výrazov a hodnôt mier spoľahlivosti systému.

Väčšinou sa dá rozlíšiť niekoľko služieb, ako aj dva alebo viac spôsobov služby, napríklad od plnej kapacity po pohotovostnú službu. Tieto režimy rozlišujú stále menej dodávok služieb. Meranie spoľahlivosti súvisiace s výkonom sa obvykle zahŕňa do pojmu výkonnosť. Na vykonávanie predpovedí spoľahlivosti z údajov o minulých zlyhaniach systému sa používajú modely rastu spoľahlivosti, či už pre hardvér, softvér alebo pre obidva typy.

Pri hodnotení systémov odolných proti poruchám má pokrytie poskytované mechanizmami na správu chýb a porúch drastický vplyv na opatrenia spoľahlivosti. Vyhodnotenie pokrytia sa môže vykonať buď modelovaním, alebo testovaním, t. j. injektovaním poruchy. Pojem spoľahlivosť a referenčná hodnota bezpečnosti, čo je postup na hodnotenie opatrení správania sa počítačového systému v prípade výskytu porúch, umožňuje integráciu rôznych techník predpovedania porúch v jednotnom rámci. Takéto kritérium umožňuje:

- charakterizáciu spoľahlivosti a bezpečnosti systému,
- porovnanie alternatívnych alebo konkurenčných riešení podľa jedného alebo viacerých atribútov.

2.2.5 Vzťahy medzi prostriedkami spoľahlivosti a bezpečnosti

Všetky návody, ktoré sa objavujú v definíciách prevencie porúch, odolnosti proti poruchám, odstraňovaní porúch a predpovedaní porúch, sú v skutočnosti cieľmi, ktoré možno zriedka, ak vôbec niekedy, úplne dosiahnuť, pretože všetky činnosti v oblasti projektovania a analýzy sú ľudskými činnosťami, tým pádom sú nedokonalé. Tieto nedokonalosti vnášajú do vzťahov, ktoré vysvetľujú, prečo spoľahlivé a bezpečné výpočtové systémy môžu najlepšie viesť iba kombinované využitie vyššie uvedených činností, najlepšie v každom kroku procesu návrhu a implementácie. Tieto vzťahy je možné načrtnúť nasledovne. Napriek predchádzaniu poruchám pomocou vývojových metodík a konštrukčných pravidiel (ktoré sú sami o sebe nedokonalé, aby boli realizovateľné), môže dôjsť k poruchám. Preto je potrebné odstránenie poruchy. Odstránenie poruchy je samo o sebe nedokonalé (nemožno nájsť všetky poruchy a pri odstraňovaní poruchy sa môžu vyskytnúť ďalšie poruchy) a bežné komponenty systému,

hardvér alebo softvér, môžu a zvyčajne aj budú, obsahovať poruchy. Z toho vyplýva dôležitosť predpovedania porúch (okrem analýzy pravdepodobných následkov prevádzkových porúch).

Zvyšujúca sa závislosť od výpočtových systémov prináša požiadavku na odolnosť proti poruchám, ktorá je založená na konštrukčných pravidlách, z tohto dôvodu je potrebné opäť aplikovať odstránenie a predpovedanie porúch na samotné mechanizmy odolné proti poruchám. Je potrebné poznamenať, že tento proces je ešte rekurzívnejší. Súčasné výpočtové systémy sú také zložité, že ich návrh a implementácia si vyžaduje softvérové a hardvérové nástroje, aby boli nákladovo efektívne (v širšom zmysle vrátane schopnosti uspieť v prijateľnom časovom rozmedzí). Tieto nástroje samotné musia byť spoľahlivé a bezpečné atď.

Predchádzajúce odôvodnenie ilustruje úzke interakcie medzi odstránením poruchy a predpovedaním poruchy a motivuje ich zhromažďovanie do analýz spoľahlivosti a bezpečnosti zameraných na dosiahnutie dôvery v schopnosť poskytovať dôveryhodnú službu, zatiaľ čo zoskupenie prevencie porúch a odolnosti proti poruchám predstavuje spoľahlivosť a zabezpečenie, zamerané na poskytovanie schopnosti poskytovať službu, ktorej je možné dôverovať. Ďalším zoskupením prostriedkov je združenie:

- prevencie a odstraňovania porúch do prevencie porúch, tzn. ako sa zameriavať na bezporuchové systémy,
- odolnosť proti poruchám a predpovedania porúch do akceptácie poruchy, tzn. ako pracovať so systémami, ktoré podliehajú poruchám.

Popri zdôraznení potreby posúdiť postupy a mechanizmy odolnosti proti poruchám, vedie zváženie odstránenia poruchy a predpovedania poruchy ako dvoch zložiek tej istej činnosti, analýzy spoľahlivosti, k lepšiemu pochopeniu pojmu pokrytie, a tým k dôležitému problému zavedenému pomocou rekurzie, hodnoteniu hodnotenia alebo ako dosiahnuť dôveru v metódy a nástroje používané na budovanie dôvery v systém. Pokrytie tu odkazuje na mieru reprezentatívnosti situácií, ktorým je systém vystavený počas svojej analýzy, v porovnaní so skutočnými situáciami, s ktorými bude systém počas svojej životnosti konfrontovaný. Pojem pokrytie je veľmi všeobecný, je možné ho spresniť uvedením rozsahu jeho použitia, napr. pokrytie softvérového testu vzhľadom na text softvéru, riadiaci graf atď., pokrytie testu integrovaného obvodu vzhľadom na chybový model, pokrytie odolnosti proti poruchám pokiaľ ide o triedu porúch, pokrytie vývojového predpokladu vzhľadom na realitu.

Posúdenie toho, či je systém skutočne spoľahlivý a ak je to vhodné, bezpečný, tzn. dodanej službe možno oprávnene dôverovať, ide nad rámec analytických techník a to minimálne z troch nasledujúcich dôvodov a obmedzení:

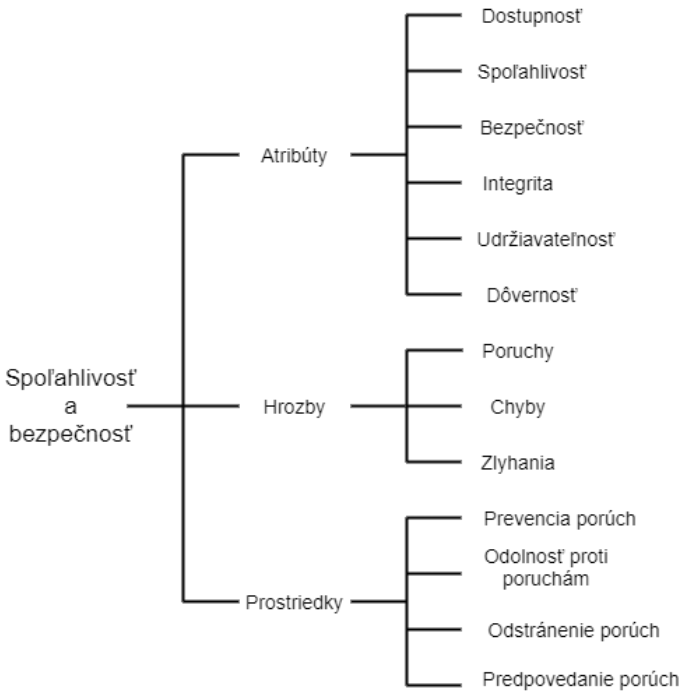
- presná kontrola pokrytia predpokladov návrhu alebo validácie s ohľadom na realitu (napr. relevantnosť kritérií použitých na stanovenie vstupov testu, hypotézy porúch pri navrhovaní mechanizmov odolných proti poruchám) by znamenala znalosť a osvojenie použitej technológie, plánované využitie systému atď., ktorá ďaleko presahuje to, čo je všeobecne dosiahnuteľné,
- vyhodnotenie systému pre niektoré atribúty spoľahlivosti a najmä bezpečnosti vzhľadom na určité triedy porúch, sa v súčasnosti považuje za nerealizovateľné alebo jeho výsledky za nevýznamné, pretože pravdepodobnostno-teoretické základy neexistujú

alebo ešte nie sú všeobecne akceptované, príkladmi sú bezpečnosť vzhľadom na náhodné poruchy vývoja, bezpečnosť pokiaľ ide o úmyselné chyby atď.,

- je pravdepodobné, že špecifikácie, s ohľadom na ktoré sa vykonáva analýza, budú obsahovať poruchy, rovnako ako akýkoľvek systém.

Tento stav má viacero následkov ako napríklad:

- pri hodnotení systému sa kladie dôraz na proces vývoja, na metódy a techniky použité pri vývoji a spôsob ich použitia, hodnotenie je v niektorých prípadoch pridelené a doručené do systému buď podľa povahy metód a techník použitých pri vývoji alebo posúdenia ich využitia,
- v špecifikáciách niektorých systémov odolných proti poruchám (okrem pravdepodobnostných požiadaviek v zmysle opatrení spoľahlivosti), prítomnosť zoznamu typov a počtu porúch, ktoré majú byť tolerované, by takáto špecifikácia nebola nevyhnutná, ak by sa dali prekonať vyššie uvedené obmedzenia (tieto špecifikácie sú v leteckých aplikáciách klasické, v podobe zoskupenia požiadaviek „fail-operating“ (FO) alebo „fail-safe“ (FS), napr. FO / FS alebo FO / FO / FS atď.



Obr. 2.5: Strom spoľahlivosti a bezpečnosti.

2.3 Vyhodnocovanie spoľahlivostných ukazateľov

Posúdenie spoľahlivosti systému musí byť založené na presne definovaných koncepciách. Je preto potrebné zhrnúť niektoré základné pojmy a potrebné definície.

Tabuľka 2.1: Základné označenie a vzorce na hodnotenie spoľahlivosti

Atribúty		
Označenie	**Vzorec**	**Význam**
$f_{X_{TTF}}(t)$	empiricky identifikované	funkcia hust. pravdepodobnosti, PDF
$f_{X_{TTR}}(t)$		
$F_{X_{TTF}}(t)$	$\int_{-\infty}^{t} f_{X_{TTF}}(x)\ dx$	kumulatívna distribučná funkcia, CDF
$F_{X_{TTR}}(t)$	$\int_{-\infty}^{t} f_{X_{TTR}}(x)\ dx$	
$R_{X_{TTF}}(t)$	$1 - F_{X_{TTF}}(t)$	spoľahlivostná funkcia
$h_{X_{TTF}}(t)$	$f_{X_{TTF}}(t)/R_{X_{TTF}}(t)$	intenzita porúch
$MTTF_{X_{TTF}}$	$\int_0^{\infty} t \times f_{X_{TTF}}(t)\ dt = \int_0^{\infty} 1 - F_{X_{TTF}}(t)\ dt$	stredná doba do zlyhania
$M_{X_{TTR}}(t)$	$\int_0^{t} f_{X_{TTR}}(s)\ ds$	udržiavateľnosť
$MTTR_{X_{TTR}}$	$\int_0^{\infty} t \times f_{X_{TTR}}(t)\ dt = \int_0^{\infty} 1 - F_{X_{TTR}}(t)\ dt$	stredná doba opravy
$MTBF_{X_{TTF}}^{X_{TTR}}$	$MTTF_{X_{TTF}} + MTTR_{X_{TTR}}$	stredná doba medzi zlyhaniami
$A_{X_{TTF}}^{X_{TTR}}(t)$	$R_{X_{TTF}}(t) + \int_0^{t} R_{X_{TTF}}(t-x)\, d(x)dx$	dostupnosť (okamžitá) kde $d(t)$ je obnovovacia funkcia hustoty
$A_{X_{TTF}}^{X_{TTR}}(\infty)$	$\lim_{t\to\infty} A_{X_{TTF}}^{X_{TTR}}(t) = \frac{MTTF_{X_{TTF}}}{MTBF_{X_{TTF}}^{X_{TTR}}}$	dostupnosť (ustálený stav)

Nech X_{ttf} je spojitá náhodná premenná predstavujúca čas do zlyhania (TTF) a zavedieme nasledujúcu notáciu fX_{TTF}(t), F_{XTTF}(t) a R_{XTTF} (t) predstavujú funkciu hustoty pravdepodobnosti (PDF), kumulatívnu distribučnú funkciu (CDF) a funkciu spoľahlivosti (skrátene spoľahlivosť) X_{TTF}. Podobne možno na zavedenie funkcií, ako je fX_{TTF}(t), F_{XTTF}(t), použiť spojitú náhodnú premennú X_{TTR}, ktorá predstavuje čas na opravu (TTR). Na základe fX_{TTF}(t) a F_{XTTR}(t) možno kvantifikovať ďalšie atribúty. Ak to z kontextu jednoznačne vyplýva, možno pre zjednodušenie zápisu vynechať „X_{TTF}“ a „X_{TTR}“. Pomocou základných výrazov a symbolov možno problém s hodnotením spoľahlivosti považovať za problém s hodnotením konkrétnych atribútov spoľahlivosti, napríklad pomocou vzorcov v tabuľke. 2.1

Analytické riešenie problému s hodnotením spoľahlivosti je známe len za určitých predpokladov. Napríklad, takýto predpoklad môže očakávať, že X_{TTF} a X_{TTR} sú vyjadrené pomocou známych rozdelení pravdepodobnosti (Exponenciálne, Normálne, Weibullovo, Gamma, atď.). Najmä pre exponenciálne distribuované X_{TTF} a X_{TTR} (parametrizované λ a μ), existuje nasledujúce analytické riešenie (používa sa zjednodušená notácia):
$f(t) = \lambda e^{-\lambda t}, F(t) = 1 - e^{-\lambda t}, R(t) = e^{-\lambda t}, h(t) = \lambda, MTTF = \frac{1}{\lambda}, M(t) = 1 - e^{-\mu t}, MTTF = \frac{1}{\mu}, A = \frac{\mu}{\mu+\lambda} + \frac{\lambda}{\mu+\lambda} e^{-(\mu+\lambda)t}$

Napriek tomu, že tieto predpoklady vyhovujú širokej škále praktických potrieb, môžu byť veľmi obmedzujúce a nedostatočné na to, aby sa ľahko (alebo vôbec) vyrovnali s niektorými faktormi. Sú však schopné kontrolovať alebo obmedzovať faktory, ako napríklad

presnosť procesu posudzovania spoľahlivosti. Tieto nedostatky potom motivujú k návrhu iných, nekonvenčných prístupov k problému posudzovania spoľahlivosti.[10]

2.4 Opravované a neopravované systémy

Z hľadiska toho, či sú systémy opravované alebo nie, ich rozdeľujeme na:

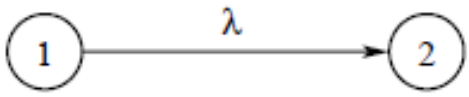
- *neopravované*
- *opravované*

2.4.1 Neopravované

Neopravované systémy sú najjednoduchšie systémy na modelovanie. Do tejto skupiny patria systémy ako simplex, duplex, triplex (TMR), alebo N-modulárny redundantný systém (NMR).

Simplex

Prvým príkladom je systém simplex. Je to systém, ktorý pozostáva z jedného počítača. Markovov model systému simplex je na obrázku 2.6.



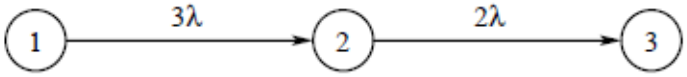
Obr. 2.6: Markovov model systému simplex[4].

V tomto Markovovom modeli stav (1) predstavuje prevádzkový stav, v ktorom simplex pracuje, stav (2) predstavuje stav zlyhania systému, v ktorom simplex zlyhal, a prechod zo stavu (1) do stavu (2) predstavuje výskyt poruchy simplexného počítača. Prechody Markovovho modelu sú exponenciálne, a preto ich možno označiť konštantnou hodnotou λ.[4]

TMR

Troj-modulárny redundantný systém (TMR) je jednou z najjednoduchších a zároveň najpoužívanejších počítačových architektúr odolných proti poruchám. Systém sa skladá z troch počítačov. Všetky tieto počítače vykonávajú úplne rovnaké výpočty s úplne rovnakými vstupmi. Počítače sa považujú za fyzicky izolované, takže počítač, ktorý zlyhal, nemôže ovplyvniť iný funkčný počítač. Matematicky sa preto predpokladá, že počítače zlyhávajú nezávisle. Ďalej sa predpokladá, že sa výstupy volia skôr, ako ich použije externý systém (tento model ich nezahŕňa), a teda sa zlyhanie nešíri svoju chybnú hodnotu do vonkajšieho sveta. Zlyhanie systému teda nenastane, pokým nezlyhajú dva počítače. Markovov model takéhoto systému je na obrázku 2.7.[4]

Počiatočný stav (1) predstavuje prevádzkový stav troch funkčných počítačov. Prechod zo stavu (1) do stavu (2) je označený 3λ, čo predstavuje mieru zlyhania niektorého z troch



Obr. 2.7: Markovov model systému TMR[4].

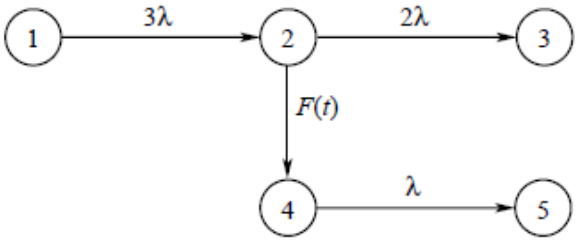
počítačov. Pretože sú všetky počítače identické, miera zlyhania λ je pre každý počítač rovnaká. Stav (2) predstavuje situáciu, kedy zlyhal jeden počítač. Prechod zo stavu (2) do stavu (3) má hodnotu 2λ, pretože v tom momente môžu zlyhať už iba dva fungujúce počítače. Stav (3) predstavuje zlyhanie systému, pretože zlyhala väčšina počítačov v systéme.[4]

2.4.2 Opravované

Systémy odolné proti poruchám sa často navrhujú pomocou rekonfigurácie. Stratégie rekonfigurácie môžu byť v rôznych variantoch, ale vždy zahŕňajú logické alebo fyzické odstránenie chybného komponentu. Techniky, ktoré sa používajú na identifikáciu chybného komponentu, a metódy, ktoré sa používajú na opravu systému, sa veľmi líšia a môžu viesť k zložitým modelom spoľahlivosti. Vyskytujú sa dve základné stratégie rekonfigurácie - degradácia a nahradenie zálohami. Metóda degradácie spočíva v trvalom odstránení chybného komponentu bez výmeny. Rekonfigurovaný systém pokračuje so zníženým počtom komponentov. Metóda záloh spočíva v odstránení chybných komponentov a ich výmene za náhradné komponenty.[4]

Degradácia triplexu do simplexu

Veľmi často využívaným príkladom degradácie je degradácia triplexu rovno do simplexu. Pri takejto degradácii sa nevyužíva postupná degradácia z triplexu do duplexu a následne z duplexu do simplexu. Markovov model degradácie triplexu do simplexu je zobrazený na obrázku 2.8.[4]



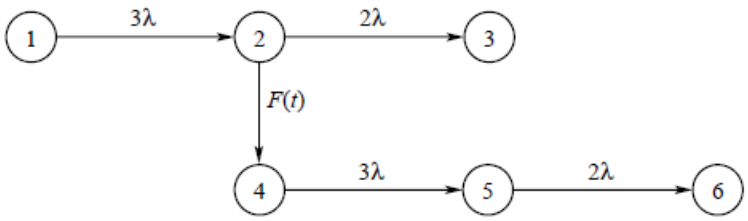
Obr. 2.8: Markovov model degradácie triplexu do simplexu[4].

Horizontálne prechody predstavujú príchody porúch. Vertikálny prechod predstavuje obnovu systému. Prechod obnovy je označený častejšie distribučnou funkciou ako mierou, ktorá naznačuje, že prechod nie je exponenciálny. Hodnota prechodu zo stavu (1) do stavu (2) je 3λ, pretože vtedy pracujú tri aktívne procesory, ktoré môžu zlyhať. Stav (2) naznačuje situáciu kedy jeden z týchto procesorov zlyhá. Pred vykonaním novej rekonfigurácie môžu zlyhať dva aktívne procesory, prechod zo stavu (2) do stavu zlyhania (3) s mierou 2λ súťaží s prechodom na zotavenie. Rekonfigurácia spočíva vo vyradení chybného procesora a jedného

z funkčných procesorov. Teda hodnota prechodu zo stavu (4) do stavu (5) je λ, pretože v aktívnej konfigurácii zostáva iba jeden procesor. [4]

TMR so zálohou

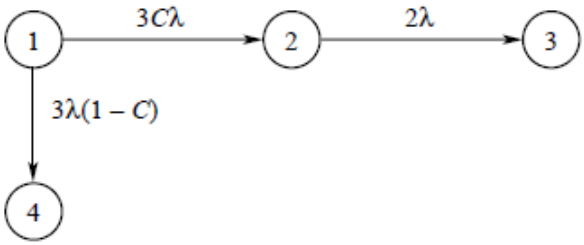
V predchádzajúcom modeli sa chybný procesor odstránil pomocou rekonfigurácie a systém pokračoval v prevádzke so zníženou úrovňou redundancie. V nasledujúcom modeli predpokladáme nahradenie chybného procesora náhradným procesorom. Opäť je využitý triplexový systém, ktorý má tentokrát jeden náhradný procesor, ktorý nezlyhá, keď je neaktívny. Model na obrázku 2.9 zobrazuje tento systém. [4]



Obr. 2.9: Markovov model systému TMR s jednou zálohou[4].

Prechod zo stavu (1) do stavu (2) má hodnotu 3λ, pretože vtedy pracujú tri aktívne procesory, ktoré môžu zlyhať. Stav (2) predstavuje dva pracujúce procesory a jeden chybný procesor. Prechod zo stavu (2) do stavu (4) predstavuje detekciu a izoláciu chybného procesora a jeho nahradenie náhradným procesorom. Pokiaľ je systém v stave (2), môžu zlyhať dva aktívne pracovné procesory, hodnota prechodu do stavu zlyhania (3) je teda 2λ. Po tom ako sa chybný procesor vymení za náhradný môžu opäť zlyhať tri aktívne procesory, hodnota prechodu zo stavu (4) do stavu (5) je teda 3λ. Tento model predpokladá, že systém sa pri ďalšom zlyhaní okamžite nedegraduje na simplex, ale pracuje ako duplex, kým ďalšia porucha neprinesie zlyhanie systému. [4]

2.4.3 Single-point zlyhania



Obr. 2.10: Markovov model systému TMR so single-point zlyhaním[4].

V predchádzajúcich modeloch sa predpokladalo, že v systéme nenastane zlyhanie jedného bodu. To spôsobí, že pri príchode jednej poruchy nastane zlyhanie systému. Ak systém nie je navrhnutý správne a je zraniteľný voči týmto zlyhaniam, spoľahlivosť sa môže vážne zhoršiť. Markovov model systému TMR so single-point zlyhaním je zobrazený na obrázku 2.10. Parameter C v tomto modeli predstavuje zlomok porúch, ktoré nespôsobujú samotné zlyhanie systému. [4]

Kapitola 3

Prostriedky pre tvorbu spoľahlivostných modelov

3.1 Časované automaty

Časované automaty (TA) sú cenným nástrojom najmä pre návrh systémov v reálnom čase. V tejto súvislosti transformujeme obvody na časované automaty, aby sme vyjadrili ich správanie v čase.[11] Časovaný automat A je šestica (L, L^0, Σ, X, I, E) kde:

- L je konečná množina miest,
- $L^0 \subseteq$ L je množina počiatočných miest,
- Σ je konečná množina symbolov (udalostí, značení),
- X je konečná množina hodín,
- I priraďuje každému miestu nejaké obmedzenie hodín z Φ(X).
  I:L $\rightarrow\Phi$(X)
- $E \subseteq$ L $\times\ \Sigma\ \times\ 2^X\ \times\ \Phi$(X) $\times$ L je množina prechodov(prepínačov). Prepínač (s,a,ϕ λ), s') reprezentuje prechod z miesta s do miesta s' symbolom a. ϕ je hodinové obmedzenie nad X, ktoré špecifikuje, či je prechod povolený a množina $\lambda \subseteq$ X definuje hodiny, ktoré majú byť vykonaním tohto prechodu vynulované.[8]

Sémantika časovaného automatu A je definovaná prechodovým systémom S_A spojeným s týmto automatom. Stav systému S_A je dvojica (s, v) taká, že s je miesto v A a v je ohodnotenie (interpretácia) hodin X také, že v spĺňuje invariant I(s). Množina všetkých stavov A sa zapisuje ako Q_A. Stav(s, v) je počiatočný stav, pokiaľ s je počiatočné miesto automatu A a platí v(x) = 0 pre všetky hodiny x $\in X.V\mathrm{S}_A$ existujú dva druhy prechodov:

- Stav systému sa môže zmeniť kvôli vypršsniu doby: pre stav (s, v) a časový prírastok v podobe reálnej hodnoty $\delta \geq 0, (s, v) \rightarrow^{\delta}$ (s, v + δ), keď pre všetky 0 $\leq\delta' \leq\delta$, v + δ' spĺňujú invariant I(s).
- Stav systému sa môže zmeniť kvôli prechodu: pre stav (s, v) a prechod (s, a, ϕ, λ), s') taký, že v spĺňuje ϕ, (s, v) $\rightarrow^a$ (s, v[λ := 0]).

Teda S_A je prechodový systém s množinou udalostí $\Sigma \cup \mathbb{R}$.

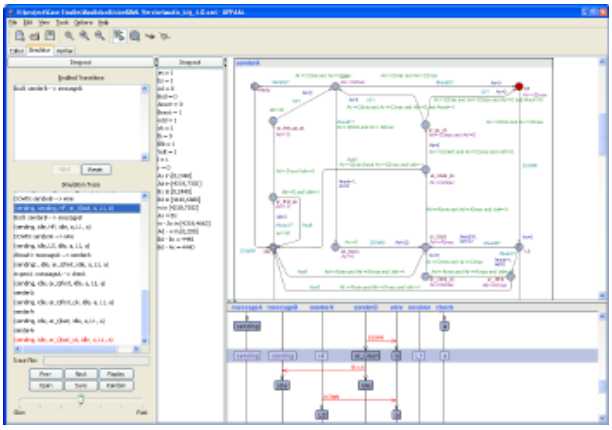
3.2 Nástroj Uppaal

Uppaal je súbor nástrojov na overovanie systémov v reálnom čase predstavovaných (sieťou) časovaných automatov rozšírených o celočíselné premenné, štruktúrované typy údajov a synchronizáciu kanálov. Tento nástroj bol spoločne vyvinutý základným výskumom v oblasti výpočtovej techniky na univerzite v Aalborgu v Dánsku a katedrou informačných technológií na univerzite v Uppsale vo Švédsku.[5]

Úspešne sa uplatňuje v prípadových štúdiách od komunikačných protokolov po multimediálne aplikácie. Je vhodný pre systémy, ktoré je možné modelovať ako súbor nedeterministických procesov s konečnou riadiacou štruktúrou a hodinami so skutočnou hodnotou, ktoré komunikujú prostredníctvom kanálov alebo zdieľaných premenných. Typické aplikačné oblasti zahŕňajú riadiace jednotky v reálnom čase a najmä komunikačné protokoly, v ktorých sú kritické aspekty časovania. Nástroj je navrhnutý na overenie systémov, ktoré možno modelovať ako siete časovaných automatov rozšírené o celočíselné premenné, štruktúrované typy údajov, funkcie definované používateľom a synchronizáciu kanálov. Obsahuje užívateľské rozhranie Java a overovací modul napísaný v C ++.[3]

Prvá verzia Uppaalu bola vydaná v roku 1995. Odvtedy je v neustálom vývoji. Experimenty a vylepšenia zahŕňajú dátové štruktúry, čiastočné zníženie objednávky, distribuovanú verziu Uppaal, riadenú a dosiahnuteľnosť s minimálnymi nákladmi, prácu na UML Statecharts, akceleráciu techniky, nové dátové štruktúry a zmenšenie pamäte. Podobnými nástrojmi ako Uppaal sú napríklad SPIN alebo TINA.

V porovnaní s konkurenčnými nástrojmi ako je napríklad SPIN je UPPAAL cenovo dostupnejší, využíva techniky na vyššiu výkonnosť a využíva nové dátové štruktúry. Táto implementácia však nie je dostatočná na zachytenie správania komplexných systémov. Problém je riešený v rámci rozšírenia UPPAAL SMC.



Obr. 3.1: Prostredie Uppaal.

3.3 Hlavné časti Uppaalu

Uppaal je zložený z troch hlavných častí, sú to:

- Systémový editor
- Simulátor

- Verifikátor

3.3.1 Systémový editor

Systém v UPPAAL je modelovaný ako súbor súbežných procesov. Vlastnosti jednotlivých procesov možno repreznetovať pomocou hodín a dátových premenných, nad ktorými sú definované dátové typy int, bool a pole integerov alebo hodnôt typu bool. Hodiny a premenné sú súčasťou takzvaných priestorov (ang. scopes), ktoré sú procesom prideľované buď lokálne alebo globálne. V rámci globálnej deklarácie možno deklarovať hodiny, konštanty, celočíselné premenné (typu integer), deklaráciu celočíselnej premennej v určitom rozsahu, 2D pole premenných typu integer alebo deklaráciu bitových polí. Syntax spomenutých deklarácií je veľmi podobná napríklad jazyku C.[3]

Pre samotný popis systému sa dajú použiť:

- Globálne deklarácie
- Šablóny procesov
- Definície systému

V rámci modelu rozlišujeme:

- *Invariant* - možno označiť ako výraz, ktorý je definovaný pomocou hodín, konštánt alebo celočíselného výrazu typu integer. V podstate sa jedná o kombináciu podmienok, ktoré sú zložené z hodín a celočíselných premenných typu integer.[3]
- *Guard* - alebo stráž, možno definovať ako určitý výraz, ktorého výsledok je rovný pravdivostnej hodnote. Jedná sa o konjukciu časových a dátových obmedzení, ktorá sa definuje na úrovni prechodov. Stráž možno zapísať ako postupnosť výrazov, ktoré sú medzi sebou oddelené čiarkami. Zápis takéhoto časového výrazu môže byť vyjadrený pomocou konštánt a premenných podľa syntaxe jazyka C. Výsledok podmienky je vyjadrený ako rozdiel hodnoty hodín, ktorý je porovnávaný s celočíselným výrazom. Guard by sme mohli definovať ako všeobecné obmedzenie vykonania hrany podmienkou.[3]
- *Update* - alebo aktualizáciu, je môžno definovať ako zoznam priraďovacích príkazov. Každý výraz je vyjadrený ako x:=e, kde x môže predstavovať premennú alebo hodiny a e celočíselný výraz. Tento výraz je vyhodnocovaný zľava doprava.[3]

Kanály, ktoré procesy využívajú na komunikáciu môžeme rozdeliť na:

- *Komunikačný kanál* - zapisuje sa pomocou chan.
- *Urgentný kanál* - zapisuje sa pomocou urgent chan. Je to príjem v čase, v ktorom je možné hranu príjemcu vykonať, prebieha bez vzájomných oneskorení Vysielač je blokovaný ak príjemca nie je pripravený na príjem.
- *Kanál pre viacsmerové vysielanie* - zapisuje sa pomocou broadcast chan. Umožňuje synchronizáciu typu 1-to-many.[1]

V rámci systému môžeme definovať aj tzv. šablónu. Šablóna možno použiť pri vytváraní nových procesov. Definuje spoločné vlastnosti procesov toho istého typu. Každý vytvorený proces sa potom stáva inštanciou danej šablóny. Šablóna môže obsahovať symbolické premenné a konštanty, ktoré môžeme nazvať aj parametre šablóny, ktoré sa používajú na vytváranie nových inštancií procesov. Súčasťou šablóny môžu byť aj lokálne hodiny a premenné, ktoré sú syntakticky rovnaké ako tie globálne. Systém je tvorený množinou procesov, ktorý tvoria tento systém. Každý proces v systéme musí byť buď:

- Proces, ktorý sa vyskytuje na ľavej strane priraďovacieho príkazu pre inštanciu procesu
- Šablóna bez parametrov

Jednotlivé typy stavov sa delia na:

- *Initial* - počiatočný v rámci daného procesu, označuje sa dvojitým krúžkom.
- *Bežné* - bežný stav v modeli.
- *Urgent* - systém strávi v danom stave 0 časových jednotiek, označuje sa písmenom U.
- *Committed* - zmena derivácie v čase na mieste je nulová, označenie písmenom C.[3]

Obr. 3.2: Jednotlivé stavy: initial, bežný, urgent, committed.

3.3.2 Simulátor

Simulátor je nástroj, ktorý sa stará hlavne o:

- Interakciu systému s užívateľom
- Sledovanie správania systému
- Vizualizáciu behu systému

Simulátor dovoľuje užívateľom pozorovať a reagovať na správanie daného systému počas realizácie. Môže sa využiť aj na vizualizáciu vykonávania generovaného verifikátorom.

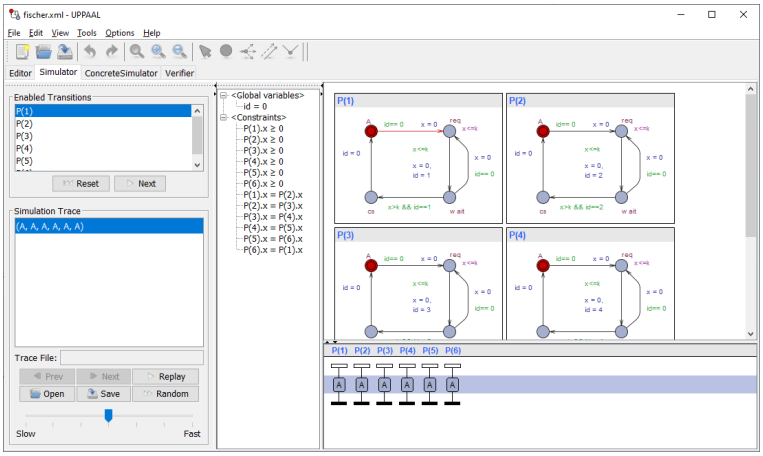
Simulátor môže byť využitý podľa troch hlavných smerov. V prvom prípade môže užívateľ spustiť systém manuálne a vybrať, s ktorým prechodom sa chce zaoberať. Druhý spôsob je, že v základnom móde môže byť nastavený tak, že systém beží od začiatku bez zásahu užívateľa. V ďalšom prípade môže užívateľ prechádzať jednotlivé kroky (uložené alebo importované z verifikátora) na zistenie ako sú jednotlivé stavy dosiahnuteľné.[3]

So simulátorom sa môže pracovať spôsobmi:

- uživateľ spustí a rozhodne, ktorý prechod bude nasledovať
- systém pobeží od začiatku do konca, bez toho aby o nasledujúcich krokoch rozhodoval uživateľ
- uživateľ zisťuje dosiahnuteľnosť jednotlivých krokov

Uživateľské rozhranie simulátora sa skladá z nasledujúcich hlavných častí:

- *Okno procesov* - zobrazuje jednotlivé inštancie procesov v danom systéme. Aktuálny stav každého procesu v každom automate má krúžok vyplnený červenou farbou a prechody, ktoré možno vykonať v aktuálnom okne, sú v každom automate označené červenou čiarou.
- *Okno premenných* - zobrazuje hodnoty premenných a hodín v aktuálnom stave alebo na zvolenom prechode.
- *Riadenie simulácie* - zabezpečuje riadenie simulácie, zároveň dovoľuje užívateľovi riadiť simuláciu a voliť stav alebo prechod, ktorý má byť vizualizovaný.
- *Záznam o činnosti systému* - postupnosť správ behu, ktorý sa práve generuje. Rozlišujú sa tu horizontálne hrany pre každý synchronizačný bod a vertikálne hrany pre každý proces.[3]



Obr. 3.3: Prostredie simulátora.[1].

3.3.3 Verifikátor

Verifikátor sa používa na overovanie invariantov a vlastností typu:

- *dosiahnuteľnosť* – umožňuje zistiť, či je možná cesta z počiatočného stavu, ktorá vedie do požadovaného stavu. Nezaručuje však korektnosť daného protokolu, ale validuje základné správanie daného modelu.
- *bezpečnosť* – informuje o zaobstaraní bezpečnosti, dôležité v kritických a nebezpečných infraštruktúrach (napr. v atómovej elektrárni)

- *živosť* – situácia kedy sa dôsledkom určitej činnosti očakáva ďalšia (napr. odoslanie a prijatie správy)

Uppaal využíva množstvo nových dotazov v rámci stochastickej interpretácie časovaných automatov. Umožňuje zobrazovanie hodnôt výrazov (prepočítaných na hodiny alebo integer) počas simulačného času. Toto udáva celkový náhľad na systém a poskytuje možnosť testovania niektorých zaujímavých vlastností daného modelu.[3]

Využíva zjednodušenú verziu jazyka CTL ako takzvaný dotazovací jazyk. Tento jazyk sa skladá zo stavových formúl a formúl definovaných v rámci prechodov medzi stavmi, ktoré tvoria cesty. Stavové formuly poukazujú na konkrétne stavy, v ktorých sa model môže nachádzať.[5]

Je možné, že nastane situácia kedy budeme chcieť spustiť väčšie a zložitejšie verifikačné úlohy. V takomto prípade nie je veľmi pohodlné spúšťať toto overovanie pomocou uživateľského rozhrania. V takom prípade je pohodlnejšie a prehľadnejšie spúšťať tieto verifikačné úlohy pomocou stand-alone príkazového riadku zvaného verifyta. Ďalšou možnosťou je spustenie verifikácie zo vzdialeného UNIX počítača. V prípade druhej možnosti sú povolené všetky argumenty, ktoré sú validné v prípade normálneho spustenia cez uživateľské rozhranie.

3.4 Rozširenie Uppaal SMC

Toto rozšírenie UPPAAL, ktoré je dostupné od verzie Uppaal 4.1, reprezentuje systémy ako sieť automatov, ktorých správanie závisí na stochastických a nelineárnych dynamických znakoch. V rozšírení UPPAAL SMC, je každý komponent systému opísaný modelom, ktorého hodiny môžu byť ohodnotené rôzne. Model v Uppaal SMC pozostáva zo siete navzájom komunikujúcich komponentov jednotlivých procesov, ktoré sú súčasťou modelu. Dva procesy môžu v rámci modelu komunikovať pomocou komunikačných kanálov a globálnych premenných.[5]

Hlavnou myšlienkou tohto rozšírenia je monitorovanie simulácií daného systému, následné využitie štatistických údajov (získaných testovaním hypotéz alebo simuláciou Monte Carlo) a rozhodnutie či daný model zodpovedá realite. SMC môžeme klasifikovať ako nástroj, ktorého miesto je niekde v strede medzi samotnými testovaním a klasickými technikami na sledovanie modelu. SMC bolo vytvorené ako implementácia menších nástrojov, ktoré boli použité na testovanie v rôznych prípadoch štúdií. Nasledovne bol tento nástroj úspešne aplikovaný v rôznych častiach vývoja a výskumu, v odvetviach ako sú biologické systémy, energetické systémy alebo v softwarovom inžinierstve s nasledovným aplikovaním v priemysle. Hlavnou príčinou úspechu tohto nástroja je, že implementácia, porozumenie a samotné používanie je jednoduché či už pre odborníkov, ktorí s daným nástrojom pracujú, ale aj pre zákazníkov, ktorí nemajú so samotným vývojom a výskumom nič spoločné. Ďalšou výhodou je využitie aj v ďalších odvetviach, odlišných od verifikácie, a to je napríklad plánovanie a robotika.[5]

Uppaal SMC je použiteľný aj na systémy, ktoré sú nedeterministické (prechody medzi stavmi sú ohodnotené nedefinovaným pravdepodobnostným rozložením), ako napríklad nástroj COSMOS, ktorý je súčasťou SMC a používa sa na hľadanie optimálnych plánovačov pre Markove procesy. V tejto práci sa budeme venovať nástroju UPPAAL SMC, ktorý sa bude zaoberať validovaním vlastností určitého deterministického modelu v danom prostredí.[5]

Okrem štandardných otázok na kontrolu modelu - t. j. dosiahnuteľnosť, invariantnosť, a nevyhnutnosť, ktoré sú stále k dispozícii - poskytuje Uppaal SMC množstvo nových dotazov týkajúcich sa stochastickej interpretácie časovaných automatov. Uppaal SMC umožňuje užívateľovi vizualizovať hodnoty výrazov (hodnotiacich celé čísla alebo hodiny) pozdĺž simulovaných behov. Toto dáva užívateľovi prehľad o správaní sa systému, aby mohol modelár požiadať o zaujímavejšie vlastnosti. Konkrétna syntax použitá v Uppaal SMC je nasledovná:[5]

$$simulate N[<= bound]E1, .., Ek \tag{3.1}$$

kde N je prirodzené číslo označujúce počet simulácií, ktoré sa majú vykonať, viazané je časové obmedzenie simulácií a E1, .., Ek sú k (stavové) výrazy, ktoré sa majú monitorovať a vizualizovať. Ako príklad môžeme uviesť príklad z tutorialu Uppaalu, kedy sa križujú vlaky (0) a vlaky (5), a aj veľkosť brány.[5]

$$simulate 1[<= 300]Train(0).Cross, Train(5).Cross, Gate.len \tag{3.2}$$

3.4.1 Syntax v Uppaal SMC

Šablóna, ktorá sa dynamicky zobrazí, musí byť deklarovaná ako dynamická šablóna. To sa deje v globálnej deklarácii modelu Uppaal pomocou dynamického kľúčového slova. Deklarácia pre šablónu klienta z tutoriálu, by bola napríklad dynamic Client (int id). Šablóna má jeden parameter id. Parametre obnoviteľných šablón sú obmedzené na parametre pass-by-value alebo odkaz na vysielací kanál. Dôvod tohto obmedzenia je, že šablóny môžu prestať existovať - zneplatnia sa všetky odkazy na jej lokálne premenné, ktoré by mohli odovzdať vloženým šablónam.[5]

Skutočné správanie použitej šablóny je definované v editore. Je však dôležité, aby medzi parametrami definovanými v dynamickej deklarácii a definíciou bola zhoda. V príklade klienta to znamená, že parametre v dynamickej deklarácii aj v definícii musia byť int id.[5]

Šablóny, ktoré sa dajú vložiť, môžu byť počas prechodu pomocou kľúčového slova na vloženie použité ľubovoľnou šablónou. Napríklad pridanie spawnu klienta (2) k aktualizačnému výrazu okraja spôsobí výskyt šablóny klienta s parametrom 2. Je zrejmé, že medzi skutočnými a formálnymi parametrami musí byť kompatibilita parametrov.[5]

Pri prechode sa môže šablóna, ktorá sa dá vytvoriť, sama roztrhnúť. Vyjadruje sa to pridaním výrazu exit() k aktualizácii okraja.[5]

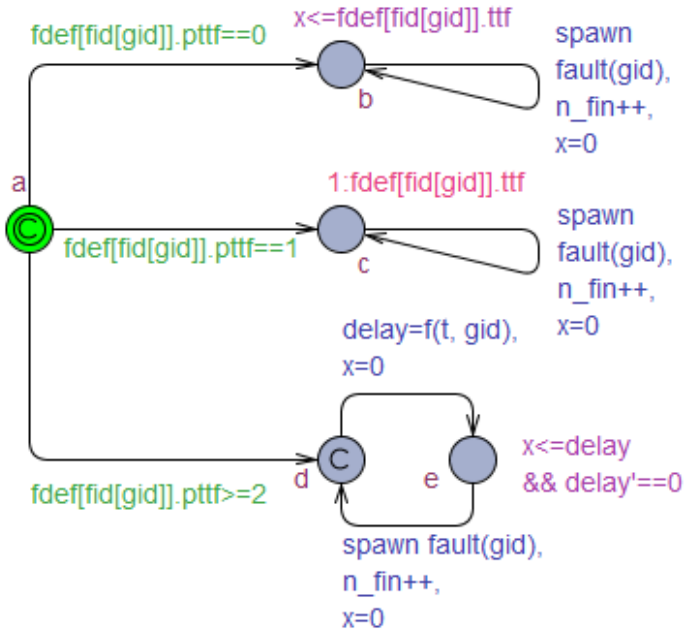
Kapitola 4

Modely systémov a experimenty

4.1 Modely v Uppaal SMC

4.1.1 Generátor porúch

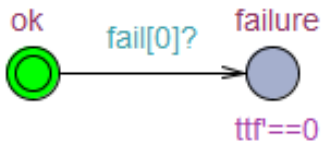
Jedným zo základných modelov je model generátora porúch. Tento generátor vytvára poruchy v časových okamihoch, ktoré sú dané rozdelením pravdepodobnosti. V čase kedy sa má porucha vyskytnúť, sa zavedie tak, že sa dynamicky vytvorí inštancia predstavujúca správanie poruchy. V prípade, že chceme vytvoriť poruchu dynamicky, je potrebné použiť funkciu spawn. Takýto spôsob modelovania sa podobá realite, po zavedení chyby do systému môže ostať v nej vopred stanovený čas a potom zmiznúť a po určitú dobu sa znova nezobrazovať (prechodná porucha), alebo sa opakovane objaviť a miznúť (prerušovaná porucha), alebo môže porucha trvať až do jej odstránenia (trvalá porucha), ktorá môže mať odlišný čas ako iné poruchy rovnakého typu.[9]



Obr. 4.1: Generátor porúch.

4.1.2 Simplex

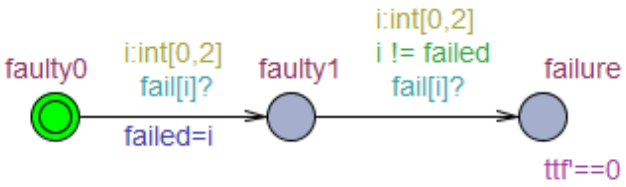
Markovov model simplexu je popísaný vyššie. Model v Uppaale je na obrázku 4.2 a vyzerá prakticko rovnako. Má len dva stavy, stav(ok), ktorý reprezentuje bežný prevádzkový stav systému a stav(failure), ktorý reprezentuje zlyhanie systému, čas ttf' sa tu rovná 0. Prechod medzi týmito stavmi je pomenovaný fail[] a reprezentuje výskyt poruchy.



Obr. 4.2: Simplex.

4.1.3 TMR

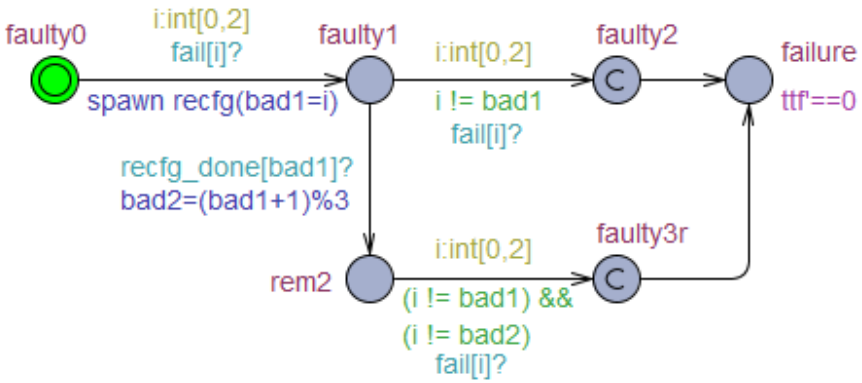
Na obrázku 4.3 možno vidieť model systému TMR. Stav(faulty0) predstavuje bežný prevádzkový stav, kedy fungujú všetky 3 počítače. Pri prechode medzi stavmi (faulty0) a (faulty1) môže zlyhať ktorýkoľvek z týchto troch počítačov. Stav(faulty1) predstavuje situáciu kedy už jeden počítač zlyhal. Pri prechode medzi stavmi(faulty1) a (failure) môže zlyhať ktorýkoľvek zo zostávajúcich dvoch počítačov. Stav(failure) predstavuje zlyhanie celého systému. Tento model sa dá považovať za východiskový pre ostatné, pretože z neho môžeme vychádzať pri tvorbe ďalších modelov. Podobne by sa postupovalo aj pri tvorbe viac N-modulárneho modelu (napríklad 5MR alebo 7MR).



Obr. 4.3: TMR.

4.1.4 Degradácia TMR do simplexu

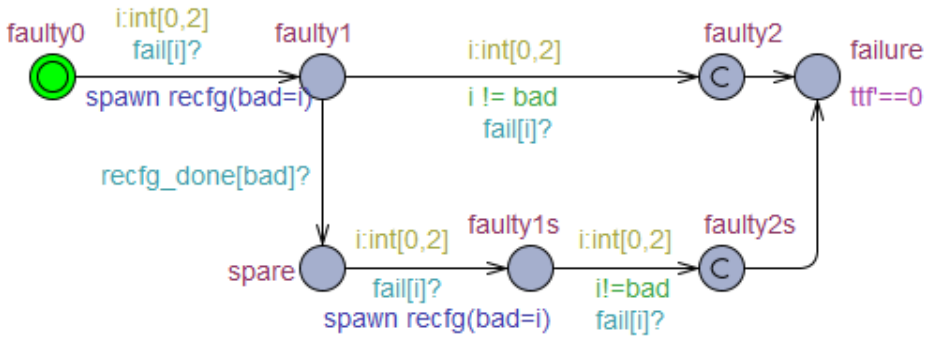
Na obrázku 4.4 je model TMR s degradáciou rovno do simplexu. Pri tomto modeli závisí degradácia od izolácie bezporuchového modulu(bad2) spolu s chybným modulom(bad1) pri prechode zo stavu(faulty1) do stavu stavu(rem2). V tomto modeli je hodnota bad2 zvolená ako nástupca bad1, pomocou operácie modulo (%3). Je možné zostrojiť aj model postupnej degradácie. V takom prípade sa systém triplex degraduje najskôr do duplexu a až potom do simplexu. Model na obrázku nám ale poskytuje rýchlejšiu cestu s menším množstvom prechodov.



Obr. 4.4: Degradácia TMR do simplexu.

4.1.5 TMR so zálohou

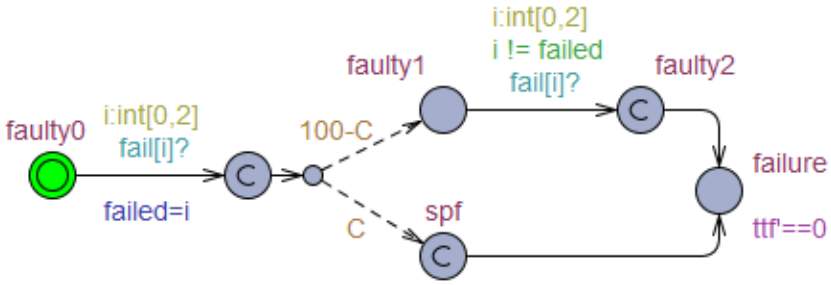
Obrázok 4.5 zobrazuje model systému TMR so zálohou. Tento model predpokladá, že keď zlyhá prvý modul(faulty1), je možné ho vymeniť za záložný. To sa deje pri prechode zo stavu(faulty1) do stavu(spare). Výmena sa ale musí dokončiť skôr, ako zlyhá druhý modul. Ak sa tak stane, sú k dispozícii opäť všetky tri funkčné moduly. V prípade, že druhý modul zlyhá skôr ako sa výmena stihne dokončiť, zlyháva celý systém.



Obr. 4.5: TMR so zálohou.

4.1.6 TMR so single-point zlyhaním

Tak ako už bolo spomínané pri Markovom modeli, v tomto systéme nastane príchodom poruchy zlyhanie systému. Systém je teda viac náchylný na celkové zlyhanie, čím je aj znížená jeho spoľahlivosť. Samotný model je podobný modelu TMR, tu však vystupuje aj parameter C, ktorý tu predstavuje malú časť porúch, ktoré nespôsobia zlyhanie celého systému.



Obr. 4.6: TMR so single-point zlyhaním.

4.2 Experimenty

Experimenty spočívali v zavedení rôznych typov porúch na jednotlivé systémy, ktoré boli predtým spomínané. Vyhodnocovala sa pravdepodobnosť toho, že systém zlyhá. Pri všetkých experimentoch boli sledované spoľahlivostné atribúty. Sledované funkcie sú vynenesené na grafoch pri každom systéme. Je to funkcia hustoty pravdepodobnosti, ktorá je na uvedených grafoch vždy vľavo. Ďalej kumulatívna distribučná funkcia, ktorá je uvedená vždy v strede a intenzita porúch (hazard rate), ktorá je uvedená vždy vpravo.

4.2.1 Popis experimentov

Experiment č.1

Prvý experiment spočínal v tom, že sa na systémy zaviedli trvalé poruchy. Tie boli vytvorené pomocou šablóny, aby sa na všetky systémy zaviedli rovnaké. Parameter TTF bol nastavený na hodnotu 200.

Experiment č.2

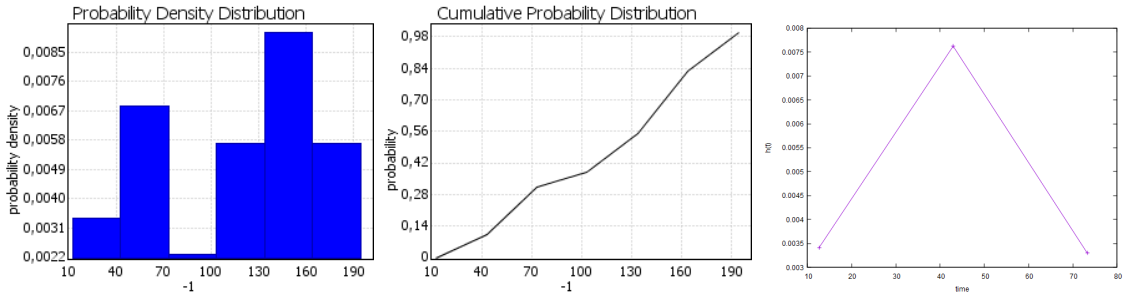
Pri druhom experimente sa na systémy zaviedli prechodné poruchy. Tie boli tiež vytvorené pomocou šablóny, takže na všetky systémy boli zavedené tie isté. Parameter TTF bol nastavený na hodnotu 200.

Experiment č.3

Pri experimente č. 3 bola na systémy zavedená kombinácia predchádzajúcich porúch. Permanentná porucha s experimentu č. 1 a dve prechodné poruchy s experimentu č.2.

4.2.2 Experiment č.1

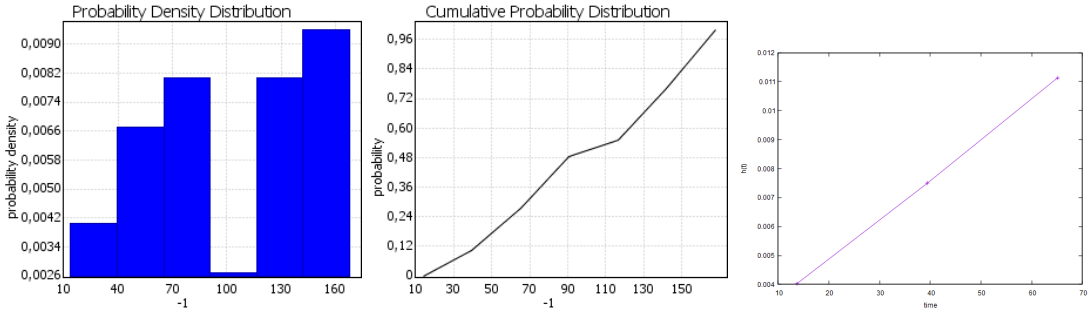
Simplex



Obr. 4.7: Simplex - experiment č.1

Simulácia prebehla veľmi rýchlo a oproti iným testom ani výrazne nezaťažila pamäť. Intenzita v tomto prípade vyšla so zaujímavým "stromovým"tvarom.

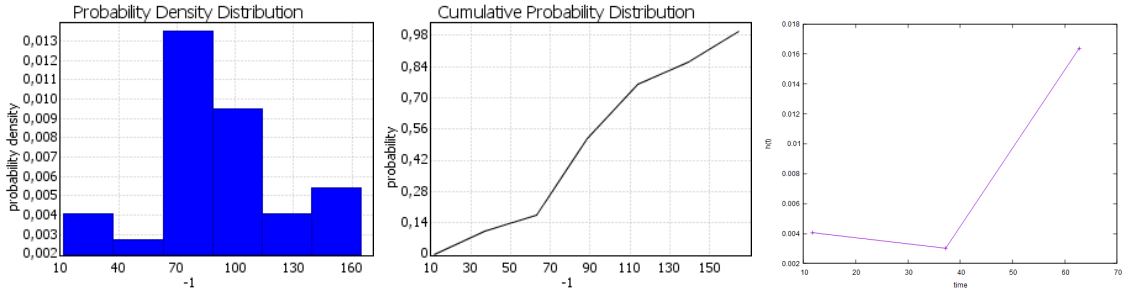
TMR



Obr. 4.8: TMR - experiment č.1

V tomto prípade simulácia zabrala takmer najvyšší čas, ale zrejme to nebolo nič čo by ju okrem toho výrazne ovplyvnilo. Náročnosť na pamäť porovnateľná, skoro rovnaká ako v prípade simplexu z predošlej simulácie. Intenzita stúpa rovnomerným spôsobom.

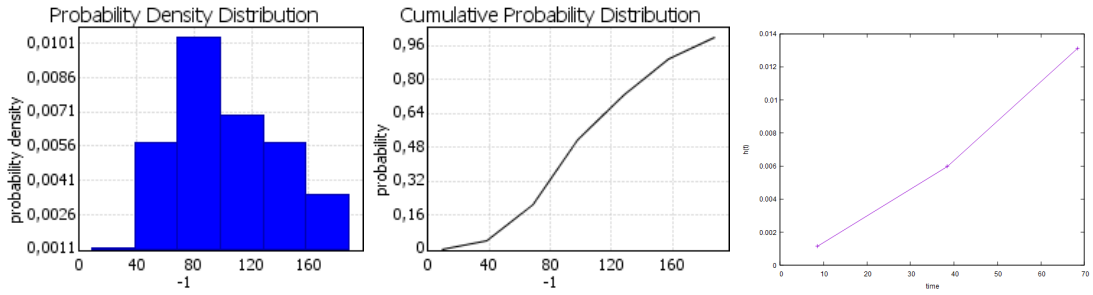
Degradácia TMR do simplexu



Obr. 4.9: Degradácia TMR do simplexu - experiment č.1

Pri degradácii TMR do simplexu sa hodnoty časovej a pamäťovej náročnosti podobali na hodnoty pri rovnakej simuláci systému simplex. V grafe inzenzity je možno vidieť výrazný skok.

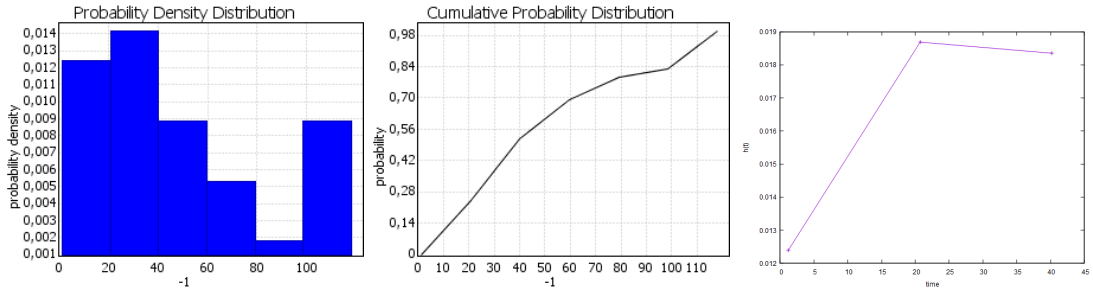
TMR so zálohou



Obr. 4.10: TMR so zálohou - experiment č.1

Časová aj pamäťová náročnosť porovnateľná ako v ostatných prípadoch. Sledované funkcie mali podobný priebeh. V grafe funkcie hustoty pravdepodobnosti vidni vplyv zapojenia zálohy.

TMR so single-point zlyhaním

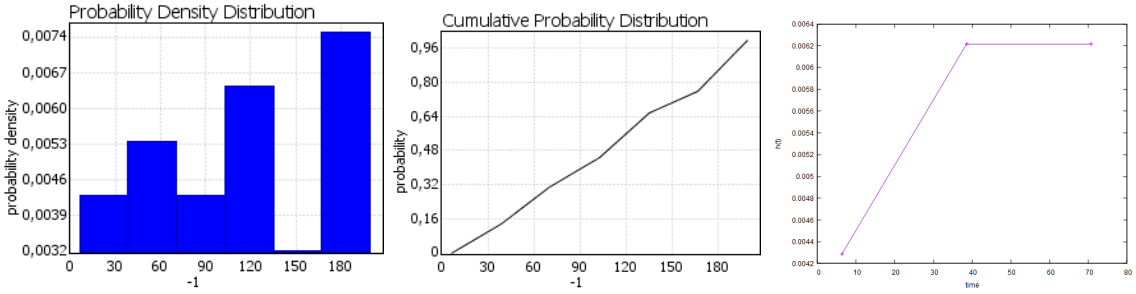


Obr. 4.11: TMR so single-point zlyhaním - experiment č.1

Náročnosť na čas bola v tomto prípade najmenšia, zatiaľ čo zaťaženie pamäte podobné s ostatnými simuláciami, v ktorých vystupovali len permanentné poruchy.

4.3 Experiment č.2

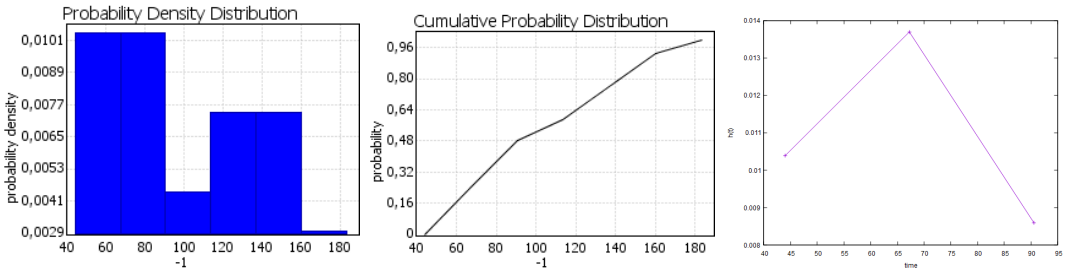
Simplex



Obr. 4.12: Simplex - experiment č.2

Pri tejto simulácii už s prechodnými poruchami výrazne stúpla časová náročnosť aj pamäťová. Pri časovej náročnosti to ale bol len ojedinelý prejav. Pamäťová naznačila, že to takto bude u všetkých simulácií s prechodnými poruchami.

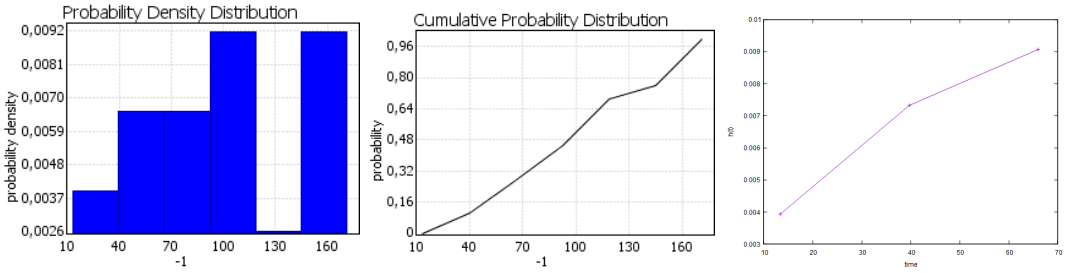
TMR



Obr. 4.13: TMR - experiment č.2

Pri TMR a prechodných poruchách sa už náročnosť času vrátila do hodnôt viac podobným tým s experimentu č. 1. Pamäťová náročnosť je ale podobná ako pri simplexe a prechodných poruchách

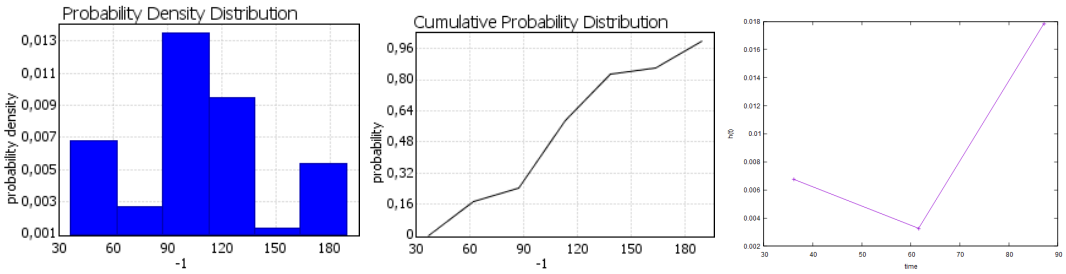
Degradácia TMR do simplexu



Obr. 4.14: Degradácia TMR do simplexu - experiment č.2

Pri degradácii TMR do simplexu sa opäť potvrdilo to čo u ostatných. Prechodné poruchy vyžadujú väčšie pamäťové zaťaženie, zatiaľ čo čas je porovnateľný so simuláciami v experimente č.1

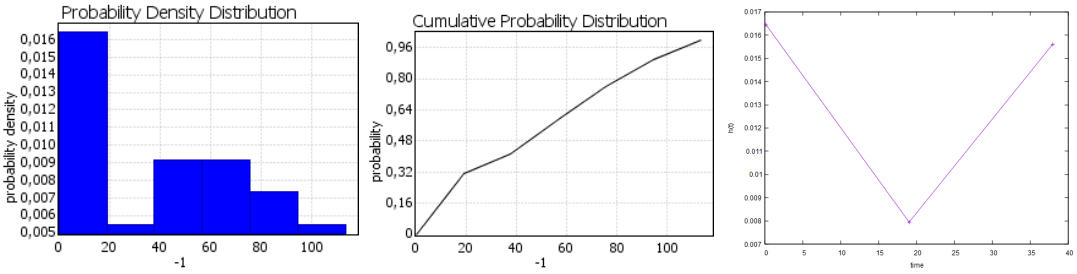
TMR so zálohou



Obr. 4.15: TMR so zálohou - experiment č.2

Simulácia TMR so zálohou si vyžadovala o niečo dlhší čas. V grafe kumulatívnej distribučnej funkcie môžeme vidieť o niečo menej hladký priebeh.

TMR so single-point zlyhaním

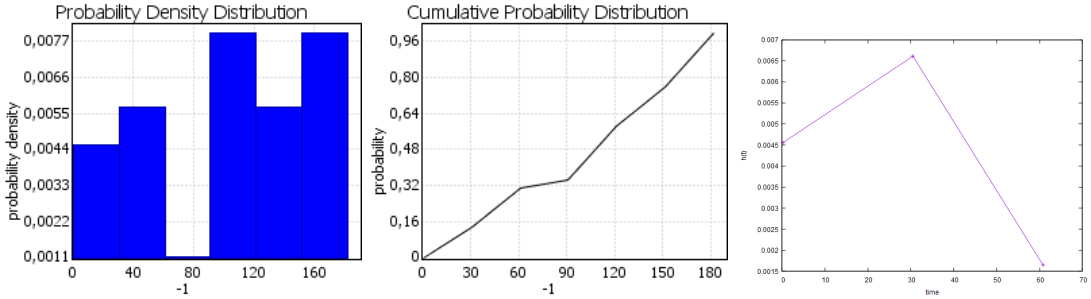


Obr. 4.16: TMR so single-point zlyhaním

Rovnako ako tomu bolo aj pri trvalých poruchách, tak aj pri prechodných, bol systém so single-point zlyhaním časovo najmenej náročný. Čo sa týka pamäťovej náročnosti, tak len potvrdil všeobecne vyššiu náročnosť prechodných porúch oproti trvalým.

4.4 Experiment č.3

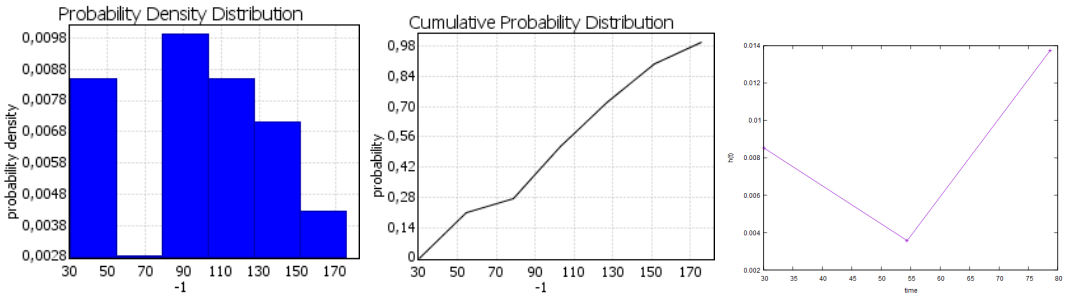
Simplex



Obr. 4.17: Simplex - experiment č.3

V grafoch simplexu možno vidieť skoky pri zavedení kombinácie porúch. Časová náročnosť tejto simulácie sa príliš nezmenila, ale očakávane narástla opäť pamäťová náročnosť.

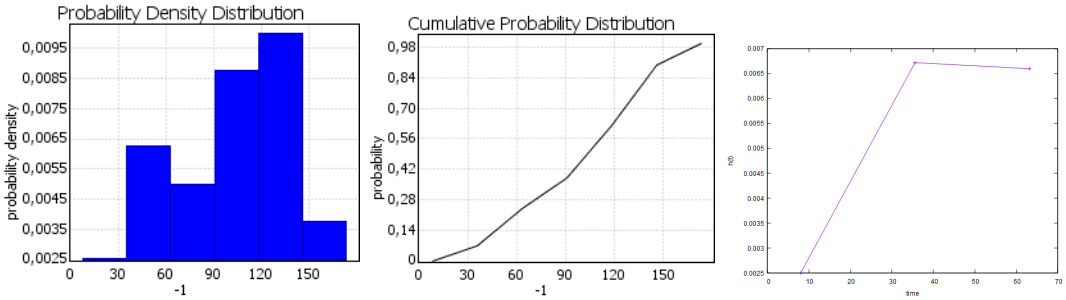
TMR



Obr. 4.18: TMR - experiment č.3

Na grafoch pri systéme TMR a kombinácii porúch môžeme vidieť výrazné skoky. Ćasová a pamäťová náročnosť podobná ako tomu bolo pri simplexe.

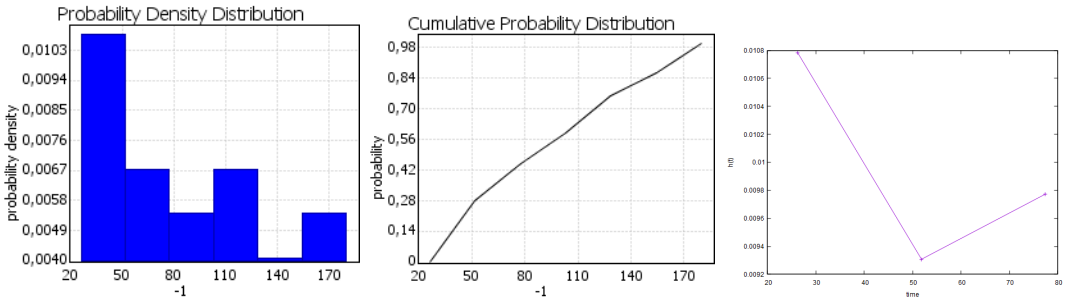
Degradácia TMR do simplexu



Obr. 4.19: Degradácia TMR do simplexu - experiment č.3

U degradácie TMR do simplexu sa potvrdil nárast pamäťovej náročnosti spolu s kombináciou porúch a podobnosť časovej náročnosti s predošlými simuláciami.

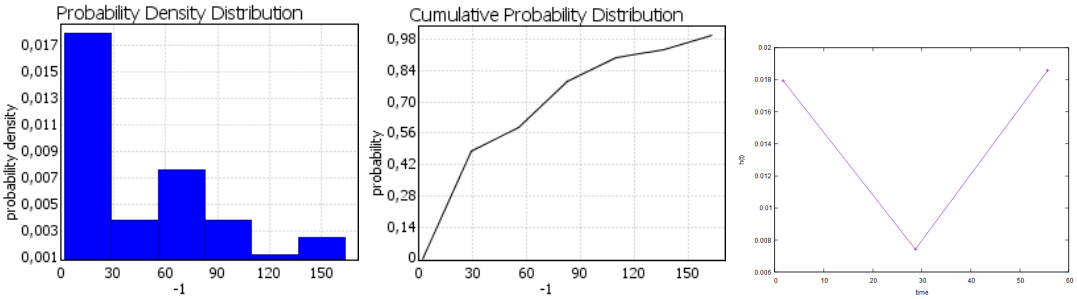
TMR so zálohou



Obr. 4.20: TMR so zálohou - experiment č.3

Kumulatívna distribučná funkcia stúpa pomene rovnomerne, zatiaľ čo v grafe intenzity môžno vidieť výrazný skok. Časová a pamäťová náročnosť ostali nezmenené.

TMR so single-point zlyhaním



Obr. 4.21: TMR so single-point zlyhaním - experiment č.3

Aj pri treťom experimente sa pri systéme TMR so single-point zlyhaním potvrdila jeho najmenšia časová náročnosť. Čo sa týka pamäťovej, tak tá bola podobná ako pri ostatných systémoch a kombinácii porúch.

4.5 Zhodnotenie

Všetky experimenty boli vykonávané za rovnakých podmienok, aj čo sa týka počtu opakovaní. Pri týchto experimentoch boli sledované spoľahlivostné ukazatele. Funkcia hustoty pravdepodobnosti a kumulatívna distribučná funkcia sa dajú zistiť priamo z Uppaalu. Intenzita porúch je spočítaná externe a grafy sú vytvorené pomocou nástroja gnuplot. Tieto funkcie je možné sledovať na grafoch pri každom systéme. Funkcia hustoty pravdepodobnosti, ktorá býva naľavo sa menila na prvý pohľad viac. Kumulatívna distribučná funkcia, ktorá býva uvedená v strede sa pri jednotlivých systémoch a odpovedajúcich poruchách menila menej. Intenzita porúch, ktorá býva vpravo, je závislá od prvých dvoch, takže pokiaľ sa

zmení jedna z prvých dvoch veličín, dá sa predpokladať aj zmena intenzity. Popri jednotlivých simuláciach bolo možné meniť štatistické parametre priamo v Uppaale, najmä teda parameter ϵ (probability uncertainty). Túto možnosť som skúsil, ale v uvedených grafoch je ponechaná všade na rovnakej hodnote a to $\epsilon = 0,05$. Popri spustených simuláciach sa dá sledovať aj náročnosť výpočtov na čas a pamäť. Čo sa týka času, tak výpočty prebiehali väčšinou v rovnakých časoch. Občas niektorý výpočet trval dlhšie, ale pripísal by som to skôr výpočtovým schopnostiam počítača a Uppaalu. Pri náročnosti na pamäť boli rozdiely už výraznejšie. Tu sa potvrdilo, že najnáročnejšie čo sa týka pamäte boli výpočty, pri ktorých sa jednalo o kombináciu porúch, tzn. že na systém boli aplikované aj prechodné aj trvalé poruchy, potom tie s prechodnými poruchami a najmenej tie s trvalými.

Kapitola 5

Záver

Cieľom tejto bakalárskej práce bolo vytvorenie spoľahlivostných modelov systémov odolných proti poruchám, aplikovanie porúch na tieto systémy a následne vyhodnocovanie spoľahlivostných ukazateľov. Na začiatok bolo nutné vyhľadať, definovať a osvojiť si základné pojmy z oblasti spoľahlivosti, systémov odolných proti poruchám a úvodu do teórie časovaných automatov. Táto práca poskytuje teoretický základ k pochopeniu danej problematiky. Pred samotným vytváraním modelov bolo nutné sa zoznámiť s nástrojom Uppaal, v ktorom sa modely tvorili. Najskôr najmä z uživateľského hľadiska, ale aj s jeho rozšírením Uppaal SMC, ktorý slúžil na simuláciu a overovanie týchto modelov. Overovanie modelov prebiehalo formou experimentov za pomoci verifátora nástroja Uppaal SMC. Experimenty spočívali v aplikovaní jednotlivých typov porúch na systémy a vyhodnocovaní atribútov, ktoré boli reprezentované pomocou grafov. Zároveň boli sledované aj časové a pamäťové náročnosti jednotlivých experimentov.

V budúcnosti by sa tieto experimenty dali určite rozšíriť o ďalšie systémy, poruchy, alebo ich rôzne kombinácie, hľadanie a overovanie konkrétnych atribútov a ich hodnôt. Prípadne pozrieť sa na to z opačného hľadiska a hľadať systémy na základe konkrétnych atribútov.

Literatúra

[1] *Uppaal help*. Uppaal, 2021 [cit. 2021-14-01]. Dostupné z: https://docs.uppaal.org/.

[2] Avizienis, A., Laprie, J.-C., Randell, B. a Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*. 2004, zv. 1, č. 1, s. 11–33. DOI: 10.1109/TDSC.2004.2. ISSN 1941-0018. Dostupné z: https://ieeexplore.ieee.org/document/1335465.

[3] Behrmann, G., David, A. a Larsen, K. G. *A Tutorial on Uppaal 4.0*. Department of Computer Science, Aalborg University, Denmark, 2006 [cit. 2021-14-01]. Dostupné z: https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf.

[4] Butler, R. a Johnson, S. Techniques for Modeling the Reliability of Fault-Tolerant Systems With the Markov State-Space Approach. Október 1995.

[5] David, A., Larsen, K. G., Legay, A., Mikučionis, M. a Poulsen, D. B. *Uppaal SMC Tutorial*. Department of Computer Science, Aalborg University, Denmark and INRIA/IRISA Rennes, France, 2018 [cit. 2021-14-01]. Dostupné z: https://www.it.uu.se/research/group/darts/papers/texts/uppaal-smc-tutorial.pdf.

[6] Liu, B. *An Efficient Approach for Diagnosability and Diagnosis of DES Based on Labeled Petri Nets: Untimed and Timed Contexts*. Dizertačná práca.

[7] Motet, G. a Powell, D. *Fault Avoidance and Fault Removal in Real-Time Systems & Fault-Tolerant Computing*. Springer, Berlin, 1999. ISBN 978-3-540-48311-3.

[8] Smrčka, A. Úvod do časovaných systému a použití pri verifikaci. 2004.

[9] Strnadel, J. On Creation and Analysis of Reliability Models by Means of Stochastic Timed Automata and Statistical Model Checking: Principle. In: Margaria, T. a Steffen, B., ed. *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*. Cham: Springer International Publishing, 2016, s. 166–181. ISBN 978-3-319-47166-2.

[10] Strnadel, J. Using Statistical Model Checking to Assess Reliability for Bathtub-Shaped Failure Rates. In: *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2019, s. 614–617. DOI: 10.23919/DATE.2019.8714878.

[11] Takan, S., Guler, B. a Ayav, T. Model Checker-Based Delay Fault Testing of Sequential Circuits. In: *ARCS 2015 - The 28th International Conference on Architecture of Computing Systems. Proceedings*. 2015, s. 1–7.

Príloha A

Obsah priloženého CD

Prehľad súborov nachádzajúcich sa na priloženom CD:

- Zdrojový kód programu
- Písomná správa vo formáte PDF
- Zdrojové súbory písomnej správy pre vytvorenie PDF súboru (latex)
- Súbor README