DEPENDABILITA INFORMAČNÍCH SYSTÉMŮ

KI/DEP

Viktor Mashkov

Ústí nad Labem 2016

Kurz:	Dependabilita informačních systémů
Obor:	Informační systémy, Informatika (dvouoborové), Informatika se zaměřením na vzdělávání.
Klíčová slova:	dependabilita, bezpečnost, diagnostika, závada, chyba, selhání
Anotace:	Kurz uvádí do problematiky dependability informačních systémů. V rámci kurzu budou podrobně vysvětleny otázky samokontroly a samodiagnostiky počítačových systémů. Kurz bude zaměřen zejména na spolehlivost a odolnost proti závadám informačních systémů.

Jazyková korektura nebyla provedena, za jazykovou stránku odpovídá autor.

Autor: Viktor Mashkov

1. Úvodní slovo
Kurz uvádí do problematiky dependability informačních systémů. V rámci kurzu budou podrobně vysvětleny otázky samokontroly a samodiagnostiky počítačových systémů.
Kurz bude zaměřen zejména na spolehlivost a odolnost informačních systémů proti závadám.
Studium se skládá ze dvou částí. Jedna část (distanční) spočívá v samostatné práci studentů. Druhá část (kontaktní) probíhá ve formě seminářů a konzultací. V průběhu samostatné práce studenti musí splnit úkoly seminářů, které jsou uvedeny v kapitolách. Pro správné řešení úkolů studenti musí prostudovat doporučenou literaturu a obsah nabízených prezentací (jsou uvedeny v každé kapitole).
V průběhu seminářů a konzultací studenti musí ukázat výsledky úkolů. Zároveň studenti budou mít možnost položit otázky a dostat vysvětlení problémů, které nezvládli pochopit.
Kurz bude ukončen zápočtem a navazující zkouškou. Pro získání zápočtu student musí doložit výsledky všech úkolů. Přitom 80% úkolů musí být řešeny správně. Po získání zápočtu student může se přihlásit ke zkoušce.
Na zkoušce studen vylosuje jednu otázku ze předem připraveného seznamu otázek. Otázky jsou vymezeny obsahem kurzu a specifikují úkoly kapitol. Seznam otázek bude zveřejněn dva týdny před zkouškou. Při hodnocení odpovědi studenta na otázku má prioritu schopnost studenta prakticky využít svoje znalosti.

Kapitola 1. Úvod do problematiky

Cíl kapitoly:
- vysvětlit vznik a význam pojmu a koncepce „Dependabilita“
- charakterizovat základní prvky Dependability informačních systémů

Klíčová slova: dependabilita, atributy dependability, služba, selhání, docílení dependability

Výkladová část:
Následující slajdy vysvětlují hlavní cíly kapitoly

Kontrolní otázky:

- vysvětlete základní prvky dependability
- charakterizujte atributy dependability
- definujte pojmy: závada, chyba a selhání

Úkoly pro samostatnou práci:
1) Prostudovat atributy dependability a prostředky pro docílení dependability informačních systémů. Použit literaturu [1], [2] a [3].
2) Vytvořit tabulku s příklady selhání různých systémů. Pro každý příklad popsat závadu, chybu a selhání systému. Tabulka má mít 10 řádků (jeden řádek pro každý příklad) a 4 sloupce. První sloupec pro popis systému, druhý pro popis závady, třetí sloupec pro popis chyby a čtvrtý sloupec pro popis selhání systému.

Kapitola 2. Odolnost informačních systémů proti závadám

Cíl kapitoly:

- vysvětlit koncepci odolnosti informačních systémů proti závadám
- podrobně rozebrat otázky odhalení chyb následujícího obnovení systému
-

Klíčová slova: odolnost proti závadám, chybný stav, selhání systému, odhalení chyb

Výkladová část:
Následující prezentace vysvětluje hlavní cíly kapitoly

Kontrolní otázky:

- co znamená odolnost systému proti závadám
- vysvětlete hlavní fáze zajištění odolnosti systému proti závadám
- charakterizujte proces obnovení systému

Úkoly pro samostatnou práci:
1) Prostudovat koncepce odolnosti systému proti závadám. Použit literaturu [1], [2] a obsah výkladové části.
2) Vytvořit tabulku s příklady odolnosti systému proti závadám. Pro každý příklad popsat odhalení chyb a obnovení systému (viz Tabulka na slajdu č. 13 ve výkladové části). Tabulka má mít 5 řádků (jeden řádek pro každý příklad) a 3 sloupce. První sloupec pro popis systému, druhý pro popis odhalení chyb, třetí sloupec pro popis obnovení systému.

Kapitola 3. Způsoby zajištění odolnosti informačních systémů proti závadám

Cíl kapitoly:

- vysvětlit hlavní způsoby zajištění odolnosti informačních systémů proti závadám
- probrat různé způsoby zajištění odolnosti informačních systémů na konkrétních příkladech

Klíčová slova: zajištění odolnosti proti závadám, obnovení, maskování, rozmanitost návrhu, opakování

Výkladová část:
Následující prezentace pomůže studentům splnit úkoly samostatné práce.

Kontrolní otázky:

- charakterizujte základní schémata zajišťující odolnost systému proti závadám
- pojmenujte a popište další schémata (sekundární) pro zajištění odolnosti systému proti závadám
- proveďte srovnávací analýzu schémat

Úkoly pro samostatnou práci:
Prostudovat způsoby zajištění odolnosti informačních systémů proti závadám. Použit literaturu [1] a obsah výkladové části.

Kapitola 4. Samokontrola a samodiagnostika na systémové úrovni

Cíl kapitoly:

- vysvětlit samokontrolu a samodiagnostiky na systémové úrovni
- procvičit návrh samodiagnostiky systémů na konkrétních příkladech

Klíčová slova: samokontrola, samodiagnostika, metody, algoritmy a organizace samokontroly a samodignostiky

Podkapitola 4.1. Podstata a základní prvky samokontroly a samodiagnostiky

Výkladová část:
Současný svět je plný složitých technických programovatelných zařízení, které řídí neméně komplexní stroje jako jsou letadla, vlaky a v poslední době i osobní automobily. Tato technická zařízení tvoří hardware a

programové vybavení, pro něž je klíčová bezchybná činnost. Důležitou roli proto hraje **kontrola** těchto zařízení.
Technická diagnostika uvažuje dvě fáze kontroly zařízení: před průběhem vlastní činnosti zařízení (*předběžná*) a v jeho průběhu (*průběžná*).
Předběžná kontrola může být prováděna pomocí *speciální kontrolní aparatur*y (SKA), která detailně prověří technický stav daného zařízení (viz Obr. 4.1) je typické především pro předstartovní diagnostiku letadel (testovací aparatura je v tomto případě rozsáhlé externí zařízení, jehož instalace přímo v letounu by byla neefektivní).

Obrázek 4.1. Speciální kontrolní aparatura

Samotná předběžná kontrola může zahrnovat vícenásobnou výměnu dat mezi SKA a zařízením a jejich následné zpracování kontrolním algoritmem v SKA. Tento kontrolní algoritmus obvykle provádí jednoduché porovnání výstupních dat ze zařízení s daty, která jsou považována za správná (etalonní).
Tato celková a mnohdy komplexní kontrola modulu, zahrnující jak výměnu dat tak provádění kontrolního algoritmu, může být na vyšší úrovni abstrakce interpretována jako jediná **atomická akce resp. kontrola** (AT = *atomic test* nebo *elementary check*). Na vyšší, tj. systémové úrovni abstrakce, jsou detaily kontroly irelevantní, v kontextu této abstrakce tudíž vystačíme s pohledem, že jeden systémový modul kontroluje druhý.
SKA je obvykle považována za bezchybnou, tj. můžeme plně důvěřovat výsledkům kontroly. V praxi je toho dosahováno různými způsoby, přičemž jedním z nejdůležitějších kritérií je úplnost kontroly. Například při kontrole programu s mnoha větvemi toku řízení (např. pro různá vstupní data) je nutno projít všemi větvemi. Pokud by byla některá větev vynechána, nebude kontrola úplná.
Hlavní výhodou SKA je proto vysoká důvěryhodnost výsledků. Naopak k nevýhodám předběžné kontroly za použití SKA patří:

- vysoké (finanční) náklady a náročné použití
- časové a režijní náklady
- nutnost zaškolení personálu (kontrola běžně vyžaduje účast operátora)
- velké prostorové nároky běžných kontrolních mechanismů.

Nyní se zaměříme na zajímavější druhou fázi, to jest **průběžnou kontrolu**. Kontrola je v tomto případě prováděna souběžně s hlavní činností zařízení. V tomto případě nelze běžně použít speciální kontrolní aparaturu (projevují se všechny výše uvedené nevýhody), její náhrada je však komplikovaná.
Jedním z řešení je použití specializovaného modulu umístěného externě, tj. vně kontrolovaného zařízení. Oproti SKA je specializovaný modul výrazně jednodušší a jeho funkce jsou omezené. Tento modul provádí kontrolu a sběr dat ve vymezených časových intervalech.
Hlavní nevýhodou použití kontrolního modulu je obtížné zajištění vysoké spolehlivosti a důvěryhodnosti výsledků kontroly. Navíc v průběhu provádění hlavní činnosti kontrolovaná zařízení navzájem komunikují, což ovlivňuje a ztěžuje jeho kontrolu.
Možným řešením výše uvedených problémů je **samodiagnostika,** tj. vzájemná kontrola a diagnostika jednotlivých účelových zařízení komplexního stroje. V tomto případě neexistuje žádné dedikované kontrolní zařízení, tj. všechna zařízení vykonávají jak běžnou tak kontrolní činnost.
Atomická kontrola může i v tomto případě zahrnovat:

1. jednoduchou kontrolu přijatých běžných dat (např. kontrolní součty)

2. složitější kontrolu přijatých dat (např. testování etalonu)
3. odesílání speciálních kontrolních dat a přijetí a zpracování odezvy.

Na systémové úrovni abstrakce odpovídá každému zařízení **modul**. Celkový model systému je tak representován grafem, v němž uzly representují moduly tj. jednotlivá dílčí zařízení a hrany atomické kontroly. Tento graf se nazývá **diagnostickým grafem** systému (zkráceně **DG**)

Každé technické zařízení (tj. modul) může být buď ve stavu, kdy poskytuje správná výstupní data (= **bezchybný modul**) resp. kdy jsou jím produkovaná data nesprávná (= modul selhal, **chybný modu**l). Bohužel může kontrolující zařízení chybně vyhodnotit data přijatá ze zařízení kontrolovaného a to v obou směrech (správná označit za chybná resp, chybná za správná) a tak zařízení nesprávně ohodnotit.

Každý modul tak má svá vlastní hodnocení modelů, které zkontroloval a tato hodnocení nemusí být konzistentní (tj. nemusí existovat konsenzus v hodnocení jednotlivých modulů). Navíc pravděpodobnost nekonzistence může být relativně velká.

Necháme prozatím stranou problémem nekonzistence v hodnoceních jednotlivých modulů, ale zamyslíme se jak je možno i přes případnou nekonzistentnost využít výsledků jednotlivých kontrol.

Na začátku si pro jednoduchost představíme, že existuje **abstraktní vnější pozorovatel**, který dostane výsledky všech dílčích atomických kontrol (viz obrázek 4.2). Předpokládejme pro jednoduchost, že přitom nedojde k žádnému chybnému přenosu nebo chybné interpretaci obdržených dat. Tento pozorovatel ve skutečnosti neexistuje, neboť kontrola musí být plně autonomní. Jeho zavedení však zjednoduší počáteční model systému.

Obrázek 4.2. Vnější pozorovatel

Lze si tak například položit otázku, zda je tento abstraktní pozorovatel schopen na základě výsledků atomických kontrol určit, které moduly jsou bezchybné a které naopak selhaly. Tento problém je již součástí systémové diagnostiky.

Diagnostika na **systémové úrovni** spočívá v odhalení všech chybných (=selhávajících) modulů. Naopak je nutno zdůraznit, že na této úrovni se neuvažuje konkrétní příčina selhání modulu (tj. co se stalo uvnitř zařízení).

V uvedeném zjednodušeném modelu je jediným vstupem diagnostiky (prováděné abstraktním vnějším pozorovatelem) množina výsledků atomických kontrol. Tato množina je označována jako **syndrom**. Výstupem je seznam chybných modulů, resp. komplementární seznam modulů chybných.

Výstup, tj. výsledek diagnostiky i jeho důvěryhodnost, závisí na několika předpokladech souvisejících s atomickými kontrolami. Nejdůležitějším je předpoklad o výsledcích kontrol prováděných jak bezchybnými tak selhávajícími moduly.

Uvažujme například atomickou kontrolu modulu M_j provedenou modulem M_i (viz obrázek 4.3).

Předpokládejme, že výsledek kontroly bezchybného modulu bezchybným modulem je roven vždy hodnotě **0**.

Obvyklá notace má tvar r_{ij} =

Ob

Složitější je situace v případě kontroly selhávajícího modulu (zde tedy např. *Mj*) modulem bezchybným (zde např. *Mi*). Většinou se předpokládá, že r_{ij} je v tomto případě rovno **1**, tj. bezchybný modul vždy odhalí chybu v modulu selhávajícím [4].
Nakonec uvážíme situaci, kdy kontrolu provádí selhávající modul. V tomto případě lze předpokládat, že výsledek bude náhodný, tj. může nabývat jak hodnoty 0 tak 1. V nejjednodušším případě budou tyto hodnoty nabývat se stejnou pravděpodobností. Existují však i další modely, například Barsi, Grandoni a Masstrini [5], nabízejí v tomto případě předpoklad, že výsledek této kontroly je vždy roven hodnoť e 1 (tj. kontrolovaný modul bude vždy označen za chybný). V praxi navíc můžeme mít i zpřesňující informace o chování selhávajícího modelu, včetně pravděpodobnosti produkování výsledků atomických kontrol (tj. např. zpřesněnou informaci o produkování zavádějících výsledků).
Všechny výše uvedené předpoklady navíc uvažují, že spojení mezi moduly jsou bezchybná (tj. při přenosu nedochází k ztrátě nebo modifikaci informací). V opačném případě by musely být předpoklady přehodnoceny. Zde však budeme vycházet pouze z následujících relativně jednoduchých předpokladů (podle Preparata).
DEFINICE 1.1:
Výsledek atomické kontroly modulu *Mj* modulem *Mi* je definován takto:

Výsledek, který poskytuje vnˇejší pozorovatel, je ovlivnˇen i strukturou atomických kontrol, které tvoˇrí *diagnostický graf.*
DEFINICE 1.2:
Syndrom *R* je uspořádaná množina výsledků jednotlivých atomických kontrol tj. $R=\{r_{ij}\}$. Jednotlivé výsledky $r_{i\ j}$ se označují jako prvky syndromu. Jednotlivé prvky syndromu mohou být v diagnostickém grafu representovány jako
ohodnocení hran, kde se hodnota rovná výsledku atomické kontroly. Viz obr. 4.4, kde ohodnocení pro přehlednost obsahuje i označení výsledků.

Obrázek 4.4. Diagnostický graf systému

Tento diagnostický graf presentuje syndrom v názorné formě. Usnadňuje provedení diagnostické analýzy (z pozice vnějšího pozorovatele. Například syndrom na obrázku 4.4 umožňuje učinit závěr, že všechny tři moduly jsou bezchybné. Samozřejmě jen tehdy, pokud platí zvolený model ohodnocení atomických kontrol.

<u>*Kontrolní otázky*</u>:
- co je atomická kontrola
- definujte diagnostický graf systému
- charakterizujte výsledek atomické kontroly

<u>*Úkoly pro samostatnou práci*</u>:
1) Prostudovat pojmy: atomická kontrola, diagnostický graf systému. Použit literaturu [1] a obsah výkladové části.
2) Vytvořit v programovacím jazyce C# objektový model reprezentující systém se samodiagnostikou. Jádrem by měly být instance třídy *Module*, implementující reprezentaci chybových stavů a metod pro atomické kontroly a objekt třídy *System*, reprezentující množinu modulů včetně diagnostického grafu. Inspiraci můžete najít v opoře „Datové struktury a algoritmy samokontroly v Pythonu“, kapitola 2.

Podkapitola 4.2. Problém určení množiny atomických kontrol. Hodnocení diagnostických grafů

Výkladová část:
Pokud je k dispozici diagnostický graf, je možno položit si otázku, zda lze systém diagnostikovat tj. určit chybné moduly, a to bez ohledu na obdržený syndrom. Lze dokázat, že úspěšnost diagnostiky závisí na počtu chybných tj. selhávajících modulů. Tento počet se označuje jako *t.* Pro některé hodnoty *t* lze vždy vytvořit diagnostický graf, který bude s úplnou jistotou zaručovat správnou diagnostiku bez ohledu na získaný syndrom.
Pro grafy, u nichž je zaručeno diagnostikování *t* chybných modulůu, tzv. **t-diagnostikovatelnost**, platí následující *nutná* podmínka:
V t-diagnostikovatelném grafu musí být modul kontrolován nejméně *t* dalšími moduly, přičemž v grafu nejsou vícenásobné hrany (= atomické kontroly).
Pokud však počet chybných modulů *t* překročí jisté t_{max} , pak již není možné takový DG zkonstruovat. Preparata ve své práci [4] dokázal, že pro toto t_{max} platí:

$$t_{max} = \quad (N-1)/2$$

kde *N* je počet uzlů resp. modulů.
t-diagnostikovatelných grafů (pro $t \cdot t_{max}$) však může existovat i více. Zajímavé jsou však především ty s malým či dokonce nejmenším počtem atomických kontrol (větší počet kontrol prodražuje a komplikuje diagnostiku)
*DEFINICE 1.3***:**
Diagnostický graf **t-optimální**, pokud obsahuje minimální počet hran, které stačí pro zajištění určité hodnoty *t* .
Pro hodnotu $t= t_{max}$ je možno graf stručně označit jako **optimální**.
Počet hran *t-optimálního grafu* lze snadno vypočítat podle následujícího vztahu:

$$l = tN$$

jenž lze v případě *optimálního diagnostického grafu* dále upravit na:

$$l = t_{max} N = \quad (N-1)/2 \quad N$$

kde
N = počet uzlů (modulů);
t_{max} = maximální hodnota parametru *t;*
l = počet hran optimálního DG.
Libovolný graf, který obsahuje více než *l* hran, je podle této definice považován za nadbytečný. Vztah lze použít i v opačném směru a pro danou hodnotu parametru *t* vytvářet instance struktur DG, které zajišťují určité diagnostické vlastnosti, například schopnost odhalit určitý počet chybných modulů. Například diagnostický graf na obr. 4.5(A) je t-optimální pro *t* = 1 neboť zajišťuje detekci pouze jednoho chybného modulu a počet hran je pro dané *t* minimální,
DG na obrázku 4.5(B) má *t* = 2 a zajišťuje tak již detekci dvou chybných modulů (a je navíc optimální neboť $t = t_{max}$ a počet hran je minimální)

Všechny diagnostické grafy zobrazené na obr. 4.6 zajišťují detekci stejného počtu chybných modulů (*t* = 1), ale mají různý počet hran. Metoda hodnocení DG navržená Preparatou neumožňuje porovnat tyto diagnostické struktury a definovat přesněji jejich diagnostické vlastnosti. Zohledňuje se pouze dosažená hodnota *t*

Obrázek 4.6.: Diagnostické grafy *s* t=1

Kromě parametru t však existují také další *kritéria* pro hodnocení diagnostických vlastností grafu, které umožňují porovnání a ohodnocení grafů na obrázku 4.6. Například každému DG může byt přiřazena hodnota pravděpodobnosti, která bude odrážet diagnostické schopnosti grafu. Konkrétně je to pravděpodobnost, že syndrom odpovídající určitému DG umožní správně diagnostikovat stav všech modulů. Tuto pravděpodobnost si vysvětlíme pomocí jednoduchého příkladu.
Nechť má například DG strukturu zobrazenou na obrázku 4.7. Z obrázku je zřejmé, že když bude bezchybný jen modul M_1

Obrázek 4.7.: Ukázkový DG pro vysvětlení pravděpodobnosti P

Pro DG na obrázku 4.7 se číslo C_1 rovná 1, protože existuje pouze jeden výběr jednoprvkové množiny uzlů, z nichž jsou ostatní uzly přímo dosažitelné (množina $\{M1\}$).
Správnou diagnostiku získáme také v případě, že jsou správné pouze dva moduly ze tří a to buď $\{M_1,M_2\}$ nebo $\{M_1,M_3\}$. Pravděpodobnost této události je rovna:

Pro uvažovaný diagnostický graf je C_2 rovno 2, neboť existují pouze dva podgrafy s dvěma uzly, z nichž jsou dosažitelné všechny ostatní uzly ($\{M_1,M_2\}$, $\{M_1,M_3$
C_3 je v tomto případě vždy rovno 1, a to bez ohledu na strukturu atomických kontrol. Výsledek diagnostiky systému je správný, pokud nastane *alespoň jedna* ze situací A_K , k = 1, . . . ,n. Z toho vyplývá, že pravděpodobnost P_{SD}

Nyní již můžeme vypočítat pravděpodobnost P_{SD} pro diagnostický graf na obr. 4.7.
Například pro P_M =

Obrázek 4.8.: Rozšířený ukázkový DG (přidány dvě hrany)

Zde byly přidány dvě hrany. Modul M_3 nyní kontroluje ostatní moduly (M_1a M_2). Pro tento DG se proto změní čísla C_1 a C_2:
C_1 = 2 výběry : {M_1, M_2}
C_2 = 3 výběry: {M_1,M_2}a {M_1,M_3} a {M_2,M_3}
Pravděpodobnost P_{SD} pro tento DG a shodnou hodnotu P_M = 0.1 je rovna (1- P_M

Nakonec uvážíme diagnostický graf z obrázku 4.9. Zde byly přidány další dvě atomické kontroly z modulu M_2. Číslo C_1 se tak zvýší na 3 (z každého uzlu lze přímo dosáhnout ostatní), Číslo C_2 zůstává na hodnotě 3. Pravděpodobnost P_{SD}

Obrázek 4.9.: Rozšířený ukázkový DG (přidány čtyři hrany)

Přidání hran zde tedy nevede ke zvýšení *t*, neboť to je rovno t_{max} . Zvýší se však pravděpodobnost získání správného výsledku diagnostiky. To lze ještě lépe vidět z grafu závislosti P_{SD} na P_M na obrázku 4.10. S rostoucí chybovostí
modulů P_M pravděpodobnost správné diagnostiky systému P_{SD} klesá, ale u DG s větším počtem atomických testů je pokles méně výrazný, Jednotlivé křivky odpovídají DG na obrázcích 4.7 (dole), 4.8 (uprostřed) a 4.9 (nahoře).
Při výpočtu pravděpodobnosti správného výsledku diagnostiky P_{SD} hrají klíčovou roli čísla C_K , která odrážejí strukturu diagnostického grafu a tudíž i strukturu atomických kontrol. Tato čísla se označují jako č*ísla charakteristická*.
DEFINICE 1.4:
Charakteristické číslo C_K, k = 1, 2, . . . ,n je *počet výběru K uzlů* (podgrafů) z diagnostického grafu, ze kterých jsou všechny *ostatní uzly přímo dosažitelné*.
U jednoduchých grafů lze charakteristická čísla zjistit snadno z nákresu diagnostického grafu. U komplexnějších diagnostických graf je však nutno využít automatizovaného výpočtu nad modifikovanou maticí sousednosti.
Modifikovaná matice sousednosti je odvozena z běžné matice sousednosti nastavením hodnoty 1 u všech prvků na diagonále (tj. je zohledněn fakt, že uzel je dostupný ze sebe sama).

Obrázek 4.10.: Funkční závislost P_{SD} na P_M a počtu AT

Při výpočtu charakteristického čísla *Ci* se vypočítává suma přes všechny řádky matice, při čemž pro každý řádek je počítán produkt přes sloupce.

Hodnota produktu pro první řádek je rovna 1, neboť a_{11} a_{12} $a_{13} = 1$. Produkt u ostatních řádků je roven 0 (a_{21} a_{22} $a_{23} = 0$, a_{31} a_{32} $a_{33} = 0$). Suma pro C_1 je tedy rovna 1.
V případě čísla C_2

Proměnné i_2 a i_1 určují jednotlivé dvojice řádků (prochází se všechny kombinace dvojic). Poté se v jednotlivých sloupcích spočítá logický součet hodnot, a to vždy jen v řádcích i_1 a i_2, a z výsledných hodnot je vypočítán produkt.

Podobně se vypočítají i další dvojice.
Program pro výpočet charakteristického čísla *Cn* pro konkrétní *n* je relativně snadný (má *n* +2 cyklů). Obecný výpočet je složitější, neboť vyžaduje generování velkého kombinací. Následující ukázka je v jazyce Haskell:

Program definuje tři pomocné funkce. Funkce *combinations* produkuje všechny n-prvkové kombinace v

podobě seznamu seznamů. Funkce *ladd* definuje logický součet nad celými čísly (resp. nad podmnožinou {0,1}). Funkce *getRow* umožňuje výběr řádků z matice (representované jako seznam seznamů). Vyjímané řádky jsou dány seznamem indexů (parametr *indices*).
Vlastní výpočet charakteristického čísla (funkce *cnum*) je již relativně přímočará. Je zde počítána suma seznamu, který je získán aplikováním dílčího výrazu na všechny *n*-prvkové kombinace řádku. Je použita tzv. seznamová komprehenze, v níž je na proměnnou *i* , která postupně odkazuje na jednotlivé kombinace aplikován výraz před svislítkem. V rámci výrazu se nejdříve provede výběr řádku podle aktuální kombinace index°u (*getRows*) a výsledek jenž je opět maticí je transponován.
Původní sloupce se tak stávají řádky a tak může být proveden výpočet logických součinů všech hodnot v řádcích pomocí mapování funkcionálu *foldr1* na jednotlivé (souvislé) řádky. *Foldr1* aplikuje binární funkci, která je uvedena jako první parametr (zde logický součet *ladd*), mezi všemi hodnotami daného řádku (tj. provede výpočet

pro dané *j*). Následně je vypočítán produkt ze seznamu výsledků této funkce pro všechny řádky (= původní sloupce).
Vyčerpávající prohledávání všech n-tic je pomalé a neefektivní, neboť výpočetní složitost je exponenciální. Částečně jej lze urychlit využitím již vypočítaných dílčích hodnot. Mnohé hodnoty jako např. řádkové součiny resp. dílčí sloupcové počty jsou v průběhu výpočtu využívány vícenásobně. Ani toto urychlení však nemusí být dostatečné v situacích, kdy je graf rozsáhlý a doba výpočtu je limitována, neboť výsledek musí být k dispozici "okamžitě". Proto se stále hledají nové metody výpočtu charakteristických čísel. Pro tyto ú˘cely lze využít různé invariantní charakteristiky
diagnostického grafu např. spektrum grafu.

<u>*Kontrolní otázky*</u>:
- co je optimální diagnostický graf
- co znamená t-diagnostikovatelnost
- co to je charakteristické číslo pro diagnostický graf

<u>*Úkoly pro samostatnou práci*</u>:
1) Prostudovat hodnocení diagnostických grafů. Použit literaturu [1] a obsah výkladové části.
2) Doplnit objektovou representaci systému se samodiagnostikou (viz úkoly v podkapitole 4.1) o metody pro hodnocení grafu (t-diagnostikovatelnost, t-optimálnost, o pravděpodobnost, že syndrom odpovídající určitému DG umožní správně diagnostikovat stav všech modulů). Při implementaci můžete vycházet z implementace v opoře „Datové struktury a algoritmy samokontroly v Pythonu“, kapitola 4 (jádrem řešení pro výpočet charakteristických čísel je iterátor přes všechny k-prvkové variace, který není na rozdíl od Pythonu v C# obsažen ve standardní knihovně)

Podkapitola 4.3. Návrh diagnostických algoritmů

<u>*Výkladová část*</u>:
Provedení atomických kontrol a získání syndromu není konečným cílem. Hlavním cílem je zjištění stavu systému, tj. nalezení všech chybných resp. správných modulů. Po provedení atomických kontrol a získání syndromu nastupuje další fáze, vlastní diagnostika systému. Tato fáze je nezbytná, pokud je výsledkem jedné nebo více atomických kontrol hodnota „1“, neboť to svědčí o přítomnosti chybných modulů v systému.
V uvažovaném modelu se předpokládá, že diagnostiku bude provádět myšlený externí pozorovatel, který získává syndrom ze systému. Externí pozorovatel není součástí systému, je sám bezchybný a bezchybný je i přenos údajů ze systému k vnějšímu pozorovateli.
Cílem samodiagnostiky je zjištění, jaký modul, resp. jaké moduly selhaly, respektive zpřesnění informace o druhu chyby. V některých případech není možné určit konkrétní chybné moduly, ale je možné pouze stanovit podmnožinu modulů, v níž jsou chybné moduly soustředěny, nicméně však může obsahovat i moduly bezchybné.
Libovolnou diagnostiku tak lze popsat jako funkci převádějící získaný syndrom na popis stavu systému $S = f(R)$, respektive jako dekódování syndromu na popis stavu (viz obrázek 4.11).
Toto dekódování je dále závislé na různých předběžných informacích o stavu systému, včetně apriorních předpokladů o chování systému v různých stavech, o vlastnostech jednotlivých atomických kontrol, nebo o režimech selhání modulů systému apod.

Obrázek 4.11. Podstata diagnostiky

Algoritmy samodiagnostiky, které se používají v praxi, zohledňují především následující dodatečné informace:

- předpoklad o výsledcích atomických kontrol prováděných jak správnými tak i selhávajícími moduly
- pořadí provádění množiny atomických kontrol
- spolehlivost jednotlivých modulů systému a spolehlivost spojení mezi moduly
- předpoklad o maximálním počtu chybných modulů. Tento předpoklad určuje mez, za kterou mohou být výsledky samodiagnostiky s určitou pravděpodobností považovány za chybné. Respektive lze na základě požadované jistoty určit mez, v jejímž rozsahu lze s danou pravděpodobností předpokládat, že zjištěný stav systému odpovídá skutečnosti.
- předpoklad o režimech selhání jednotlivých modulů systému. Většinou je uvažováno, zda jsou selhání jednotlivých modulů řízená či nikoliv. Selhání modulů mohou být například stálá, přechodná nebo nahodilá.
- možností obnovení systému (tj. nahrazení chybného modulu nebo jeho průběžné odstranění).

První obecné algoritmy samodiagnostiky začaly být navrhovány počínaje rokem 1967, kdy byl publikován Preparatův článek [4]. Od té doby byla navržena celá řada dalších algoritmů samodiagnostiky. Následující přehled se věnuje těm nejzákladnějším a nejužitečnějším.

Algoritmy založené na tabulce syndromu

Tabulkové algoritmy pracují s **tabulkou syndromu**, což je maticová representace syndromu. Tabulka syndromu $M_R[r_{ij}]$ je čtvercová matice o rozměru $N \times N$, kde N je počet modulů. Pokud je součástí syndromu výsledek atomické kontroly, kterou provádí i-tý modul na modulu j-tém, to jest hodnota r_{ij}, pak tabulka syndromu obsahuje tuto hodnotu v i-tém řádku a j-tém sloupci. Položka v tomto případě obsahuje hodnotu nula nebo jedna.
Pokud není atomická kontrola mezi určitými dvěma moduly systému provedena (tj. není v syndromu k dispozici), pak je tato situace graficky representována pomlčkou v průsečíku příslušného sloupce a řádku. Při representaci tabulky syndromu v počítači lze použít například hodnotu -1.

Obrázek 4.12.: Dvě varianty representace syndromu

Pokud je graf t-diagnostikovatelný, lze pomocí tabulkových algoritmů identifikovat všechny chybné moduly, ale samozřejmě pouze v případě, že jejich počet nepřekročí hodnotu t. V opačném případě bude algoritmus s velkou pravděpodobností schopen tuto situaci detekovat, ale chybné moduly nemohou být identifikovány (tj. program může nanejvýše signalizovat, že počet chybných modulů je příliš velký). Výsledkem však může být i zcela zmatečná identifikace chybných modulů.
Přípravným krokem tabulkových algoritmů je proto určení hodnoty t z diagnostického grafu. Běžnější je však využití ad hoc navrženého diagnostického grafu se zaručenou hodnotou t, která je obvykle zároveň optimální, tj. je rovno $t_{max} = \lfloor (N-1)/2 \rfloor$.
Před volbou a použitím tabulkového algoritmu je navíc nutné zohlednit vlastnosti atomických kontrol. Většina tabulkových algoritmů pracuje s vlastnostmi AT podle definice Preparata [4] .

Při volbě konkrétního tabulkového algoritmu je rozhodující počet modulů v systému, neboť tyto algoritmy jsou na počtu modulů silně závislé. Větší počet modulů může algoritmus výrazně zkomplikovat, a tak může být čas provedení diagnostiky pro větší počet modulů neakceptovatelný (např. s exponenciální časovou složitostí), resp. výrazně závislý na konkrétním syndromu. Další vývoj se proto zaměřuje na návrh tabulkových algoritmů s akceptovatelnou a predikovatelnou časovou složitostí.

Algoritmy založené na tabulce potenciálních syndromů

Dalším příkladem tabulkových algoritmů jsou algoritmy založené na *tabulce potenciálních syndromů*. Jeden z prvních algoritmů tohoto typu byl navržen Vedeshenkovem [6].

Tabulka potenciálních syndromů se vytváří před začátkem diagnostické procedury. Podkladem pro vytvoření tabulky je matice sousednosti diagnostického grafu a reprezentace vlastností atomické kontroly (většinou se volí opět reprezentace podle definice Preparata).

Tabulka potenciálních syndromů zahrnuje všechny syndromy, které mohou být obdrženy pro různé přípustné kombinace stavů modulů. Zpravidla se opět předpokládá, že počet chybných modulů v systému nepřekročí hodnotu *t*. V tomto případě stačí uvažovat jen ty kombinace stavů, v nichž je počet chybných modulů menší než *t*

Tak například pro systém, jehož diagnostický graf je zobrazen na Obr. 4.12, budou uvažovány pouze následující situace:

S_1: M_1 je chybný
S_2: M_2 je chybný
S_3: M_3 je chybný
S_4: M_4 je chybný
S_5: M_5 je chybný
S_6: M_1 a M_2 jsou chybné
S_7: M_1 a M_3 jsou chybné
S_8: M_1 a M_4 jsou chybné
S_9: M_1 a M_5 jsou chybné
S_{10}: M_2 a M_3 jsou chybné
S_{11}: M_2 a M_4 jsou chybné
S_{12}: M_2 a M_5 jsou chybné
S_{13}: M_3 a M_4 jsou chybné
S_{14}: M_3 a M_5 jsou chybné
S_{15}: M_4 a M_5 jsou chybné

Když skutečný stav systému neodpovídá žádné z uvažovaných situací (například v případě, když je počet chybných modulů větší než dva) je výsledek diagnostiky nesprávný nebo dokonce zavádějící. Diagnostické algoritmy, které používají tabulky potenciálních syndromů, proto mají jen omezenou důvěryhodnost. Tuto důvěryhodnost však lze předem spočítat.

Tabulka potenciálních syndromů má následující sloupce:

- označení uvažované situace (S_i)
- čísla chybný modulů $\{M_j\}$, $j = 1…n$ pro danou situace S_i
- označení potenciálního syndromu (R^i_p) pro danou situaci S_i
- jednotlivý prvky syndromu $\{r_{ij}\}$ pro danou situaci, tyto prvky tvoří l sloupců, kde l je počet atomických kontrol (resp. hran).

Tabulka potenciálních syndromů obsahuje tolik řádků, kolik je uvažovaných situací. Hodnoty jednotlivých prvků potenciálního syndromu $\{r_{ij}\}$ jsou stanoveny podle zvolené reprezentace atomické kontroly. V případě Preparatovy representace mohou jednotlivé prvky nabývat hodnot 0,1 nebo X v závislosti na stavech modulů M_i a M_j. Hodnota označovaná jako X vyjadřuje náhodný výsledek kontroly prováděné chybným modulem. Ve skutečném syndromu může nabývat hodnoty 0 nebo 1 s určitým náhodným rozdělením. Pravděpodobnostní charakteristiky atomických kontrol nejsou pro tento algoritmus podstatné.

Pro systém s diagnostickým grafem zobrazeným na obrázku 4.12 je tabulka potenciálních syndromů následující:

Jak lze z tabulky snadno vidět, kterékoli dva potenciální syndromy (tj. kterékoli dva řádky v tabulce) jsou odlišné alespoň v jednom prvku. Tato odlišnost syndromů je klíčová pro diagnostiku na základě potenciálních syndromů.
Vlastní algoritmus je prováděn po skončení běhu atomických kontrol, to jest po získání skutečného syndromu. Algoritmus spočívá v porovnání skutečného syndromu se syndromy potenciálními. Cílem je nalézt potenciální syndrom, který odpovídá syndromu reálnému, což umožní identifikovat chybné moduly (jsou uvedeny v druhém sloupci tabulky). Při porovnání se musí shodovat všechny výsledky atomických kontrol, nejednoznačný výsledek u potenciálního syndromu (označený jako „x“) se shoduje s libovolným výsledkem reálné kontroly (žolíkové porovnávání).
Pro porovnávání existuje několik strategií. Základní a nejjednodušší strategie, jež spočívá v postupném porovnávání reálného syndromu s jednotlivými řádky tabulky, je neefektivní, neboť vyžaduje největší počet porovnání (maximálně až $Q \cdot l$).
V našem ukázkovém případě, v němž je reálný syndrom roven $R_A = \{r_{12}=0, r_{13}=1, r_{23}=1, r_{24}=0, r_{34}=1, r_{35}=0, ,r_{45}=1, ,r_{41}=0, ,r_{51}=1, r_{52}=1\}$, lze i při použití základní strategie snadno nalézt shodující se potenciální syndrom r_p^{14} a tím diagnostikovat stav modulů v systému (chybné jsou moduly M_3, M_5, správné M_1, M_2, M_4). I při ručním prohledávání tabulky se však jako výhodnější jeví postupný předvýběr řádků pomocí několika prvních hodnot syndromu (například, pokud zohledníme jen první prvek syndromu, omezí se výběr na 11 potenciálních řádků).
Tento algoritmus předvýběrů lze snadno rozšířit a implementovat. Nejdříve jsou vybrány řádky, u nichž se shoduje první prvek s reálným syndromem (jsou to řádky 1, 3, 4, 5, 6, 7, 8, 9, 13, 14, 15). V druhém kroku se zaměříme jen na vybrané řádky a testujeme shodu u druhého prvku syndromu. Výběr se opět omezí na řádky (1, 3, 6, 7, 8, 9, 13, 14). Postupný výběr pokračuje a končí v sedmém kroku, v němž se rozhodne mezi variantami potenciálních syndromů r_p^3 a r_p^{14}. Počet porovnání je v tomto případě výrazně menší [B].
Na závěr této sekce shrneme některé přednosti a nevýhody tabulkových algoritmů plynoucí z jejich návrhu i použití:
výhody:

- potřebují pouze základní informace o systému (ty máme zpravidla vždy k dispozici)
- tabulky mohou být snadno vytvořeny a jsou názorné, což snižuje chybovost při jejich zpracování

nevýhody:

- tabulkové algoritmy nezohledňují spolehlivost jednotlivých modulů systému, což snižuje důvěryhodnost výsledku

Pravděpodobnostní algoritmy
Pravděpodobnostní algoritmy samodiagnostiky jsou zaměřeny na výpočet aposteriorní pravděpodobností stavů jednotlivých modulů systému. Po zjištění daných pravděpodobností může být učiněno rozhodnutí buď o stavech všech modulů v systému, nebo alespoň o některých modulech (v případě nedostatku informací). Důvěryhodnost výsledků může být navíc zvýšena zohledněním předběžných informacích o spolehlivosti jednotlivých modulů. Z tohoto důvodu nejsou pravděpodobnostní algoritmy omezeny jen na *t*-diagnostikovatelné grafy s maximálně *t* chybnými moduly, ale mohou poskytovat relevantní informace i při nižším počtu atomických kontrol, nebo při větším počtu chybných modulů.
K návrhu pravděpodobnostních algoritmů přispěli především H. Fujiwara a K. Kinoshita, kteří již v roce 1981 navrhli jednoduchý a efektivní algoritmus [7].
Nejdříve si připomeňme, že pro zjištění stavu modulu můžeme použít jak *apriorní* tak *aposteriorní* pravděpodobnost. *Apriori* znamená doslova „před“. Proto přívlastek apriorní vyjadřuje pravděpodobnost určitého stavu modulu ještě před provedením atomických kontrol. Tato pravděpodobnost je ve většině případů stanovena na základě dodatečných informací, například informace o spolehlivosti modulu. Tato informace bývá uváděna v dokumentaci k modulu. Běžně to bývá např. parametr λ, tj. intenzita exponenciálního rozdělení selhání. Apriorní pravděpodobnosti se mohou u jednotlivých modulů lišit. Tím se pravděpodobnostní přístup k diagnostice liší od přístupu tabulkového, který vychází z předpokladu, že všechny moduly mají stejnou (apriorní) pravděpodobnost selhání. Zahrnutím specifického chování jednotlivých modulů se může zvýšit důvěryhodnost diagnostiky, neboť ta již například nemusí být omezena hodnotou *t*.
Aposteriorní pravděpodobnost naproti tomu vyjadřuje pravděpodobnost určitého stavu modulu po provedení jeho kontroly. Je zřejmé, že aposteriorní pravděpodobnost správného stavu u správného modulu je vyšší než apriorní, neboť výsledky kontroly přidávají další informace o stavu modulů, čímž zvyšují naši jistotu o stavu systému.
Základem algoritmu je výpočet aposteriorní pravděpodobnosti jednotlivých stavů (správný/chybný) u všech modulů v systému. Pro tento účel využijeme jednoduchý příklad diagnostického grafu s třemi moduly (M_1, M_2, M_3) a třemi atomickými kontrolami. Diagnostický graf je znázorněn na obrázku 4.13.

Obrázek 4.13.: Ukázkový DG a syndrom pro popis pravděpodobnostního algoritmu

Výpočet aposteriorních pravděpodobností stavů modulů začíná výpočtem aposteriorní pravděpodobností hypotéz všech možných stavů modulů. Zde je pouze poněkud zjednodušen model apriorních pravděpodobností a tím zkrácena symbolika.
Předpokládejme, že apriorní pravděpodobnosti bezchybného stavu modulů jsou známé a nabývají hodnoty P_1,P_2 a P_3. Symboly q_1, q_2 a q_3 označují apriorní pravděpodobnost, že moduly jsou v chybném stavu. Je zřejmé, že $q_1 = 1 - P_1$, $q_2 = 1 - P_2$ a $q_3 = 1 - P_3$. Dále předpokládejme syndrom podle obrázku 4.13.
1. určení všech možných hypotéz ohledně stavu modulů. V našem případě se všechny moduly mohou nacházet jak ve stavu správném tak chybném. Je tak nutno uvažovat osm hypotéz:

H_1:	123	M_1 je správný	M_2 je správný	M_3 je správný
H_2:	123	M_1 je správný	M_2 je správný	M_3 je chybný
H_3:	123	M_1 je správný	M_2 je chybný	M_3 je správný
H_4:	123	M_1 je správný	M_2 je chybný	M_3 je chybný
H_5:	123	M_1 je chybný	M_2 je správný	M_3 je správný
H_6:	123	M_1 je chybný	M_2 je správný	M_3 je chybný
H_7:	123	M_1 je chybný	M_2 je chybný	M_3 je správný
H_8:	123	M_1 je chybný	M_2 je chybný	M_3 je chybný

2. výpočet pravděpodobnosti všech hypotéz
$P(H_1) = P_1P_2P_3$
$P(H_2) = P_1P_2q_3$
$P(H_3) = P_1q_2P_3$
$P(H_4) = P_1q_2q_3$
$P(H_5) = q_1P_2P_3$
$P(H_6) = q_1P_2q_3$
$P(H_7) = q_1q_2P_3$
$P(H_8) = q_1q_2q_3$
Protože hypotézy $H_1 \ldots H_8$ popisují všechny možné situace, je jejich celková pravděpodobnost rovna jedné.
3. určení podmíněných pravděpodobností
Výpočet provádíme pro událost, v níž atomické kontroly τ_{12}, τ_{23}, τ_{31} skončí s výsledkem (syndromem) podle obrázku 4.13. Proto musíme nejdříve vypočítat pravděpodobnost získání tohoto syndromu za podmínky, že stavy modulů odpovídají určité hypotéze. Jednotlivé podmíněné pravděpodobnosti i zde závisí na representaci výsledků atomických kontrol. Pokud využijeme klasickou representaci Preparatovu a navíc budeme předpokládat, že pravděpodobnost jedničkového výsledku je vždy $P_r = P\{X = 1\}$ (tj. výsledek kontroly prováděné chybným modulem nezávisí na stavu kontrolovaného modulu), získáme následující pravděpodobnosti:

$P(R/H_1) = 0$	$P(R/H_5) = 0$
$P(R/H_2) = 1 - P_r$	$P(R/H_6) = (1 - P_r)^2$
$P(R/H_3) = 0$	$P(R/H_7) = 0$
$P(R/H_4) = 0$	$P(R/H_1) = (1 - P_r)^2P_r$

Nulové pravděpodobnosti jsou u hypotetických stavů, které nemohou daný syndrom produkovat. Například, pokud by byly všechny moduly správné (hypotéza H

$_1$), pak by nemohl správný modul M_2 označit za chybný modul M_3 (výsledek atomické kontroly r_{23} je jedna). Hypotéza H_6 je naproti tomu slučitelná se syndromem a pravděpodobnost je součinem tří nezávislých pravděpodobností: $P(R{:}r_{12} = 0 / H_6)$, což je pravděpodobnost, že chybný modul M_1 (viz hypotéza) provede kontrolu modulu M_2 s výsledkem „1“ $= (1 - P_r)$; $P(R{:}r_{23} = 1 / H_6) = 1$, neboť správný modul vždy odhalí chybný; a nakonec $P(R{:}r_{31} = 0 / H_6) = (1 - P_r)$ ze stejných důvodů jako výše (hypotéza předpokládá, že M_3 je chybný). Pravděpodobnosti hypotéz H_2 a H_8 lze vyjádřit obdobným způsobem.

4. určení podmíněné pravděpodobnosti hypotéz při daném syndromu

Pro výpočet jednotlivých podmíněných pravděpodobností $P(H_i / R)$ lze využít Bayesův vztah. Jmenovatel zlomku vyjadřuje pravděpodobnost syndromu R při daných apriorních pravděpodobnostech, tj. $P(R) = \sum^{\ell}_{i=1} P(H_i)P(R / H_i)$. Pro náš příklad je $P(R)$ rovno $(1 - P_r)P_1P_2q_3 + (1 - P_r)^2q_1P_2q_3 + (1 - P_r)^2P_rq_1q_2q_3$.
Pravděpodobnosti jednotlivých hypotéz za podmínky získání daného syndromu mají následující tvar (nulové jsou vynechány):

5. určení aposteriorní pravděpodobnosti správnosti jednotlivých modulů

Aposteriorní pravděpodobnost, že je určitý modul správný, lze získat z podmíněných pravděpodobností jednotlivých hypotéz. Tato pravděpodobnost je totiž rovna podmíněné pravděpodobnosti události, v níž stav systému odpovídá libovolné hypotéze, která daný modul považuje za správný. Například v našem případě je modul M_2 považován za správný v hypotézách H_1, H_2, H_5 a H_6 .
Podmíněnou pravděpodobnost sjednocení hypotéz lze získat součtem podmíněných pravděpodobností těchto hypotéz (podmínka je ve všech případech stejná, po provedení atomické kontroly je získán určitý syndrom).
Když aposteriorní pravděpodobnost správnosti modulu M_i označíme symbolem P^*_i

V našem případě má pro modul M_2

Pro názornost můžeme aposteriorní pravděpodobnost vyčíslit pro konkrétní hodnoty apriorních pravděpodobností například pro $P_1 = P_2 = 0.8$. Dále předpokládejme, že pravděpodobnost P_r je rovna 0.5 (chybný modul vrací při kontrole ostatních modulů se stejnou pravděpodobností buď hodnotu „0“ nebo „1“).
Podle uvedeného vztahu se aposteriorní pravděpodobnost správnosti modulu M_2 rovná 0.986.
Jak lze vidět, po provedení trojice atomických kontrol se zvýšila naše jistota o správnosti modulu z 0.8 (apriorní pravděpodobnost) na téměř 0.99, neboť atomické kontroly byly v souladu s apriorním předpokladem.
Dílčí pravděpodobnosti a výsledky pro všechny moduly ukazuje obrázek 4.14 (získaný z tabulkového kalkulátoru *OpenOffice Calc*

Obrázek 4.14.: Výpočet aposteriorní pravděpodobnosti správnosti modulů

Výpočet aposteriorních pravděpodobností správnosti jednotlivých modulů je však pouze podkladem vlastní diagnostiky. Hlavním cílem je stejně jako u výše uvedených diagnostických algoritmů identifikace správných i chybných modulů.

Kontrolní otázky:
- vysvětlete podstatu diagnostiky na základě množiny výsledků atomických kontrol
- charakterizujte algoritmy založené na tabulce syndromu
- charakterizujte pravděpodobnostní algoritmy

Úkoly pro samostatnou práci:
1) Prostudovat návrh diagnostických algoritmů. Použit literaturu [1] a obsah výkladové části.
2) Implementovat tabulkový a pravděpodobnostní algoritmus v modelu systému se samokontrolou (viz úkoly v podkapitole 4.1). Vzorem mohou být implementace v opoře „Datové struktury a algoritmy samokontroly v Pythonu“, kapitoly 7,8).

Podkapitola 4.4. Diagnostika intermitentních selhání

Výkladová část:
Intermitentní selhání modulů má na rozdíl od selhání permanentních, tj. trvalých, občasný resp. přerušovaný charakter. Důvody takového chování mohou být různé a představují specifickou oblast výzkumu. Zde pouze poznamenejme, že jednou z příčin může být například vliv radiace.
Pro naše účely jsou mnohem zajímavější matematické modely intermitentně selhávajících modulů. Jeden z takových modelů byl navržen Mallelem a Massonem [8]. Tento model uvažuje dva stavy intermitentního selhání, a to stav pasivní P_s a stav aktivní A_s. Dále používá dva číselné parametry λ a μ určující přechody mezi těmito stavy (viz obr.4.15).

Obrázek 4.15. Model intermitentních selhání

Tento i další modely se používají k modelování a testování metod diagnostiky intermitentních selhání.
Pro diagnostiku intermitentních selhání je možno použít metody používané u stálých selhání (viz výše), je však nutno uvažovat tři stavy modulů: správný, permanentní selhání a intermitentní selhání. U pravděpodobnostního algoritmu je tak např. nutno uvažovat 3^n hypotéz, což může být výpočetně velmi náročné (a to i v případě malých systémů, neboť např. 3^{10} je řádově rovno 10^{10}). Kromě toho může použití pravděpodobnostních algoritmů vést k situaci, kdy budou mít dvě hypotézy o stavu modulu stejnou resp. podobnou aposteriorní pravděpodobnost, což by vyžadovalo další kritéria pro rozhodnutí. Tato situace může vzniknout především v případě stejných apriorních pravděpodobností u jednotlivých modulů.

Obrázek 4.16. Systém s intermitentními selháními

Získaný syndrom je kompatibilní s výše uvedeným předpokladem o stavu modulů. Pokud však máme k dispozici pouze tento syndrom, nelze učinit rozhodnutí, který z modulů M_1 a M_3 je správný a který má intermitentní selhání.
Nalezení obecného intermitentního selhání je velmi obtížné, neboť charakter selhání (vyjádřitelný například pomocí parametrů λ a μ) se může výrazně měnit. Pro některé druhy intermitentních selhání však existují speciální metody, které nalezení a identifikaci chybných modulů výrazně usnadňují.
Navíc je nutno zdůraznit, že v případě intermitentních selhání je důležitá nejen vlastní identifikace selhávajícího modulu, ale i stanovení dalšího postupu pro zacházení s tímto modulem. I zde existují specifické třídy selhání, včetně intermitentní selhání, u nichž je vysoká pravděpodobnost, že modul může být používán i nadále bez jakéhokoliv druhu opravy.
Na základě modelování intermitentních selhání modulů s parametry λ a μ a se zohledněním časových okamžiků atomických kontrol t_{AT} lze intermitentní selhání rozdělit do následujících tří druhů:
1.druh zahrnuje intermitentní selhání, která mohou být odhalena při několika málo (2 až M) opakovaných atomických kontrolách
2.druh intermitentní selhání, která sice mohou být odhalena po opakovaných kontrolách, avšak těchto kontrol však musí být relativně velký počet (až např. v řádu 10^6)
3.druh intermitentní selhání, u nichž je možnost zachycení velmi nepravděpodobná, resp. zachycení nastává nejvýše jedenkrát během diagnostiky.
Souběh intermitentních selhání a atomických kontrol lze nejlépe znázornit na modifikovaném sekvenčním diagramu, jenž je znám například z modelovacího jazyka UML. Ukázkový diagram je na obrázku 4.17.

Obrázek 4.17. Sekvenční diagram intermitentního systému

Jednotlivé moduly jsou v tomto diagramu zobrazeny jako svislé čáry. Ve směru ze shora dolů roste čas. Atomické kontroly jsou znázorněny jako vodorovné čáry se šipkami. Jejich vertikální poloha (osa y) určuje čas provedení atomické kontroly, přičemž atomická kontrola ležící výše je provedena před atomickou kontrolu ležící níže. Počátek a konec šipky (v horizontálním směru) určuje modul provádějící kontrolu (počátek) a modul kontrolovaný (konec). Pod šipkou je zapsán obdržený syndrom (syndromy na obrázku odpovídají definice Preparata [4], kde X je náhodná veličina).
Pro příklad, první (nejčasnější) atomická kontrola je prováděna modulem M_1 a kontrolován je modul M_4. Výsledkem může být jednička nebo nula.
Selhání je v grafu znázorněno tmavě šedým obdélníkem u svislice daného modulu, jenž pokrývá časové období, v němž modul selhává. Modul M_1 je permanentně chybný po celou dobu diagnostiky systému, u modulu M_2 se také jedná o selhání permanentní, jenž vzniká až v průběhu diagnostiky systému a diagnostiku nijak neovlivňuje, neboť modul se po selhání již neúčastní atomických kontrol. Modul M_3 selže jen na velmi krátký okamžik a pouze jednou. Jedná se tedy o intermitentní selhání třetího druhu (je zachyceno jedenkrát). U modul M_4 se opět jedná o intermitentní selhání, které se však opakuje a výrazněji ovlivňuje diagnostiku. Díky malému počtu atomických kontrol nelze stanovit, zda se jedná o selhání prvého nebo druhého druhu.
Pro diagnostiku intermitentních selhání prvého druhu byly navrženy metody založené na sumárním syndromu R_Σ, jenž může být získán po M-násobně opakovaném provedení množiny atomických kontrol.
Sumární syndrom spočítáme takto:

$$R_\Sigma = \{r^*_{ij}\},\ r^*_{ij} = \bigvee_l r^l_{ij},\ \text{kde } r^l_{ij} \in R_l \text{(syndrom obdržený při } l\text{-tém opakování)}$$

Operace „$\vee$“ je zobecněný logický součet. Jinak řečeno, pokud bude v jednom opakování AT zjištěn u atomické kontroly výsledek „0“ a ve druhém opakování „1“, je sumární výsledek roven $0 \vee 1 = 1$ a modul je označen za chybný.
Diagnostika může být provedena pouze v případě, že bude splněna následující podmínka:

$$R_\Sigma \in R_o$$

kde R

$_o$ je množina sumárních syndromů, které mohou být obdrženy v případě, že jsou možná pouze trvalá selhání modulů, za podmínky, že počet nesprávných modulů nepřekročí hodnotu *t*.
Pokud je podmínka splněna, je provedena běžná diagnostika např, pomocí tabulkové metody, ale vychází se ze sumárního syndromu. Moduly označené za chybné mají buď permanentní selhání, nebo se jedná o intermitentní selhání prvého druhu.
Podmínka není splněna, je-li výsledkem sumární syndrom, který je nekonzistentní tj. obsahuje výsledky, které jsou konfliktní. Výsledek je konfliktní, pokud je v sumárním pohledu některý modul jedním správným modulem označen za správný a druhým taktéž správným za chybný. V tomto případě se obvykle další diagnostika neprovádí a výsledkem je jednoduché oznámení, že systém nemůže být správně diagnostikován. Tento případ nastává v případě, kdy v systému existují moduly s intermitentními selháními druhého a třetího druhu.
Pokud jsou však k dispozici další prostředky (časové a režijní), může procedura diagnostiky dále pokračovat, čímž lze získat další informace. Nejjednodušší možností rozšíření je zvýšení počtu opakování množiny atomických kontrol. Poté mohou být některé intermitentní moduly odhaleny jako selhávající, a to i z pohledu modulů, které tuto skutečnost ještě neodhalily, což vede k odstranění nekonzistencí a zařazení těchto modulů mezi moduly se selháním prvého druhu.
Druhá možnost je o něco zajímavější, neboť sice vyžaduje složitější prozkoumání sumárního syndromu a přináší i určitý risk (tj. pravděpodobnost nesprávného výsledku diagnostiky), nevyžaduje však provádění dalších kontrol.
Základem této metody je předpoklad, že všechna zbývající neodhalená intermitentní selhání jsou třetího druhu Pravděpodobnost chybného výsledku je rovna pravděpodobnosti nesplnění tohoto předpokladu. Tento předpoklad je odůvodněný, neboť v reálných složitých systémech se tento typ intermitentních selhání vyskytuje mnohem častěji než ostatní druhy selhání.
V tomto případě (tj. pokud není splněna podmínka, je prvním krokem stanovení podmnožiny *Z*, do níž patří všechny moduly, které mohou být na základě sumárního syndromu R_Σ označeny jako správné. V druhém kroku je nutno ověřit konzistentnost všech výsledků atomických kontrol, jež jsou prováděny moduly z podmnožiny *Z*. Je tedy nutno zjistit, zda moduly z této podmnožiny stejně hodnotí moduly z podmnožiny doplňkové ($\overline{Z}$) nebo nikoliv. V průběhu zjišťování může nastat jedna ze situací znázorněných na obrázku4.18.
Situace A znázorněná na obrázku 4.18 může vzniknout z následujících příčin:

1. modul M_j selhal v okamžiku těsně před provedením poslední atomické kontroly v posledním opakování AT (zde je to kontrola provedená modulem M_i tj. τ_{ij})
2. modul M_j má intermitentní selhání druhého druhu a modul M_i je jediný z modulů, který jej zaregistroval
3. modul M_i permanentně selhal. Atomická kontrola τ_{ij} je první kontrola, která byla tímto selháním dotčena
4. modulu M_i má intermitentní selhání. Selhání bylo zachyceno pouze atomickou kontrolou τ_{ij}.
5. buď modul M_i nebo M_j má intermitentní selhání třetího druhu. Toto selhání se projevilo v průběhu atomické kontroly τ_{ij}.

Současné intermitentní selhání obou modulů nebudeme uvažovat, neboť pravděpodobnost takovéto události je velmi nízká.

Obrázek 4.18.: Situace způsobené intermitentním selháním třetího druhu

Jednotlivé možné příčiny konfliktů jsou přehledně znázorněny v sekvenčních diagramech na obrázku4.19. V případech 2, 4, 5 existuje více možných souběhů selhání a atomických kontrol. Znázorněn je však vždy pouze jeden, resp. u pátého případu výjimečně dva.

Obrázek 4.19.: Souběhy selhání a atomických kontrol v situaci A

V souladu s přijatým předpokladem budeme dále uvažovat pouze intermitentní selhání třetího druhu (viz příčina 5).
Můžeme proto tvrdit, že buď modul M_i nebo M_j má krátkodobé intermitentní selhání. Takové selhání se s vysokou pravděpodobností nebude opakovat a modul bude nadále fungovat bez problémů. Proto není pro další úvahy důležité, který z obou modelů selhal. Jediným cílem je odstranění nekonzistencí v syndromu. Řešení je v tomto případě snadné, stačí zaměnit výsledek kontroly τ_{ij} z hodnoty „1“ na hodnotu „0“ .
Situace B z obrázku 4.18 je snadněji řešitelná, neboť může nastat pouze v případě, že modul M_j má intermitentní selhání, které odhalily všechny moduly z podmnožiny Z vyjma modulu M_i. Ukázky možných souběhů jsou na obrázku4.20. Řešení je jednoznačné, stačí zaměnit výsledek kontroly τ_{ij}

Obrázek 4.20. Souběhy selhání a atomických kontrol v situaci B

Komplikovanější situace vzniká v případě nekonzistence u dvouprvkové podmnožiny správných modulů Z

Obrázek 4.21. Situace způsobená intermitentním selháním třetího druhu (spec. případ)

Obdržený výsledek lze interpretovat buď jako situaci A (a řešit ji změnou jednoho ohodnocení na „0“) nebo jako situaci B (v tomto případě jsou ohodnocení sjednocena na „1“). Abychom mohli učinit rozhodnutí a přiklonit se k jednomu z řešení, musíme porovnat pravděpodobnosti obou situací. V případě interpretace podle situace A by se jednalo o intermitentní selhání třetího druhu modulu M_i nebo M_j. Podle druhé interpretace (situace B) musí mít modul M_i selhání druhého druhu. Protože pravděpodobnost vzniku selhání tohoto druhu je menší, je vhodnější zvolit řešení podle situace A, tj. sjednotit syndrom na hodnotu „0“ ($\tau_{ij} = 0$, volíme pozitivnější řešení).
Na konci podkapitoly ještě shrneme specifické rysy systémů s intermitentními selháními:
I. Hlavním cílem diagnostiky není identifikace modulu s intermitentním selháním, ale řešení konfliktních situací, které vznikají z důvodů specifického charakteru těchto selhání.
II. Samotná diagnostika zahrnuje několik dílčích kroků:
Krok 1: vícenásobné opakování množiny atomických kontrol a získání sumárního syndromu R_{Σ}.
Krok 2: ověření, zda obdržený syndrom odpovídá podmínce R

$_{\Sigma} \in R_o$. Pokud je tato podmínka splněna, pak diagnostika probíhá stejně jako v případě permanentních selhání. V opačném případě se přechází k dalšímu kroku.
Krok 3: učení množiny *Z*, jenž obsahuje moduly, které lze na základě sumárního syndromu jednoznačně považovat za správné.
Krok 4**:** kontrola konzistentnosti výsledků atomických kontrol prováděných moduly z množiny *Z*.
Krok 5**:** vyřešení konfliktních situací.
III. Intermitentní selhání je možno rozdělit do tří druhů podle počtu opakování množiny atomických kontrol nutných pro zachycení selhání modulu.
Intermitentní selhání prvého druhu jsou odhalena již v kroku 2. Selhání druhého druhu však mohou být odhalena až po provedení kroku 3 (a to pouze některá). Selhání třetího druhu jsou tolerována. To znamená, že systém je schopen dalšího provozu bez vnějšího zásahu. Konfliktní situace způsobené intermitentními selháními třetího druhu jsou řešeny v kroku 5.
IV. Nevýhodou diagnostiky intermitentních selhání je výrazně vyšší časová složitost resp. režie systému. Proto může být velmi náročné, ne-li nemožné provádět tuto diagnostiku v průběhu provozu reálných systémů.

Kontrolní otázky:
- popište model intermitentních selhání
- charakterizujte různé druhy intermitentních selhání
- co je to sumární syndrom
- jak se provádí diagnostika intermitentních selhání

Úkoly pro samostatnou práci:
1) Prostudovat diagnostiku intermitentních selhání. Použit literaturu [1] a obsah výkladové části.
2) Seznámit se s UML nástroji pro sekvenční diagramy a prakticky ukázat jejich možnosti pro popis diagnostiky v systémech s intermitentními selháními (podle representace použité v obrázku 4.17). Použít můžete například *Enterprise Architect*, dostupný pro studenty KI).

Kapitola 5. N-variantní programování a objektové orientované programování

Cíl kapitoly:
- vysvětlit výhody použití objektového programování pro N-variantní programování
- probrat specifiku použití objektů v N-variantním programování

Klíčová slova: NVP, rozmanitost, manažer, adjudikátor, varianta, obnovení, formální model

Výkladová část:
Následující prezentace vysvětluje hlavní cíly kapitoly

Kontrolní otázky:

- vysvětlete specifiku použití objektů pro N-variantní programování
- popište funkce manažera a adjudikátoru
- vysvětlete, jak probíhá obnovení systému

Úkoly pro samostatnou práci:
1) Prostudovat použití OOP pro N-variantní programování. Použit literaturu [1] a obsah výkladové části.
2) Vytvořit v C# jednoduchou knihovnu pro podporu n-variantního programování za využití generik a návrhového vzoru Command (s využitím generických rozhraní resp. delegátů).

Kapitola 6. Ošetření výjimek pro N-variantní programování

Cíl kapitoly:

- vysvětlit model ošetření výjimek
- podrobně rozebrat interní a externí výjimky
- probrat otázku adjudikace výjimek

Klíčová slova: výjimka, interní a externí výjimky, ošetření výjimek, adjudikace výjimek

Výkladová část:
Následující prezentace pomůže studentům splnit úkoly samostatné práce.

Kontrolní otázky:

- popište model ošetření výjimek v NVP
- charakterizujte externí výjimky
- jak probíhá adjudikace výjimek
- popište výjimkové podmínky (s použitím příkladů)

Úkoly pro samostatnou práci:
1) Prostudovat ošetření výjimek v N-variantním programování. Použit literaturu [1] a obsah výkladové části.
2) V programovacím jazyce C# vytvořit program pro ošetření výjimek v NVP.

Kapitola 7. Konkurenční a spolupracující souběžné systémy

Cíl kapitoly:

- charakterizovat tři kategorie souběžných systémů
- podrobně rozebrat kooperační a konkurenční souběžnost
- probrat metody slučující kooperační a konkurenční souběžnost

Klíčová slova: souběžnost, aktivní komponent, pasivní komponent, transakce, AKIT vlastnosti

Výkladová část:
Následující prezentace pomůže studentům splnit úkoly samostatné práce.

Kontrolní otázky:

- definujte tři kategorie souběžnosti
- definujte co je to aktivní a pasivní komponent systému
- pojmenujte vlastnosti atomické transakce
- popište multi-vláknovou transakci

Úkoly pro samostatnou práci:
1) Prostudovat kooperační a konkurenční souběžnost. Použit literaturu [1] a obsah výkladové části.
2) Prostudovat a na praktickém příkladě presentovat implementaci transakcí na úrovni jazyka C# (třída *TransactionScope*). Vyzkoušejte je pro implementaci transakcí mezi dvěma databázemi.

Kapitola 8. Konverzace v distribuovaných systémech

Cíl kapitoly:

- charakterizovat synchronizační problém u distribuovaných systémů
- popsat hlavní metody umožňující řešit tento problém

Klíčová slova: konverzace, synchronizace, zpráva, souběžnost výstupu, validace

Výkladová část:
Následující prezentace vysvětluje hlavní cíly kapitoly

Kontrolní otázky:

- charakterizujte synchronizační problém u DS
- charakterizujte metodu „Dívat se Dopředu“
  popište metodu, která používá rozesílání informace o vstupu
- co znamená virtuální synchronizace
- vysvětlete, co znamená souběžnost výstupu

Úkoly pro samostatnou práci:
Prostudovat konverzace v distribuovaných systémech. Použit prezentaci ve výkladové části.

Kapitola 9. Koordinované atomické činnosti

Cíl kapitoly:

- vysvětlit co je Koordinovaná Atomická Akce (KAA)
- probrat hlavní problémy návrhu KAA
- uvést příklady praktického využití KAA

Klíčová slova: vlastnosti KAA, vnoření, konkretizace, ošetření výjimek

Výkladová část:
Následující prezentace vysvětluje hlavní cíly kapitoly

Kontrolní otázky:

- definujte problémy návrhu KAA
- jak probíhá ošetření výjimek v KAA
- vysvětlete problém dezerce vláken
- popište jak může být zajištěna odolnost KAA proti závadám

Úkoly pro samostatnou práci:
Prostudovat koordinované atomické akce. Použit prezentaci ve výkladové části.

Kapitola 10. Dependabilita distribuovaných aplikací

Cíl kapitoly:

- vysvětlit jak může být docílena dependabilita distribuovaných aplikací
- probrat hlavní moduly architektury distribuovaných aplikací

Klíčová slova: průhlednostní vlastnosti, modul distribuovaného systému, atomická akce, dálkové volání procedury, trvalý objekt

Výkladová část:
Následující prezentace vysvětluje hlavní cíly kapitoly

Kontrolní otázky:

- charakterizujte základní předpoklady pro distribuovaný systém
- popište dvoufázový commit protokol
- charakterizujte hlavní moduly distribuovaného systému

Úkoly pro samostatnou práci:
Prostudovat odolnost distribuovaných systémů proti závadám. Použit prezentaci ve výkladové části.

Kapitola 11. Použití skupin objektů pro zajištění odolnosti proti závadám

Cíl kapitoly:

- vysvětlit použití replikace objektu pro zajištění odolnosti proti závadám
- probrat tři aspekty řízení konzistenci replik
- vysvětlit replikační politiky

Klíčová slova: replika, řízení konzistenci replik, aktivace objektu, replikační politiky

Výkladová část:
Následující prezentace vysvětluje hlavní cíly kapitoly

Kontrolní otázky:

- v čem spočívá řízení konzistenci replik
- definujte a popište replikační politiky
- vysvětlete, jak probíhá aktivace objektu

Úkoly pro samostatnou práci:
Prostudovat použití skupin objektů pro zajištění odolnosti proti závadám. Použit prezentaci ve výkladové části.

Kapitola 12. Dependabilita s ohledem na závady způsobené svévolnými činnostmi

Cíl kapitoly:

- charakterizovat politiky bezpečnosti
- probrat bezpečnostní vlastnosti
- probrat bezpečnostní selhání
- vysvětlit metody zajišťující bezpečnost systému

Klíčová slova: politiky bezpečnosti, bezpečnostní vlastnosti, bezpečnostní selhání, intrusion (tj. nežádoucí vniknutí), útok, zranitelnost

Výkladová část:
Následující prezentace pomůže studentům splnit úkoly samostatné práce.

Kontrolní otázky:

- charakterizujte politiky bezpečnosti
- definujte bezpečnostní vlastnosti
- charakterizujte bezpečnostní selhání
- definujte intrusion, útok a zranitelnost

Úkoly pro samostatnou práci:
1) Prostudovat politiky bezpečnosti, bezpečnostní vlastnosti, bezpečnostní selhání a metody zajišťující bezpečnost. Použit prezentaci ve výkladové části a literaturu [1].
2) Vytvořit tabulku s příklady bezpečnostních selhání různých systémů. Pro každý příklad popsat závadu (intrusion, útok, zranitelnost), chybu a selhání systému. Tabulka má mít 5 řádků (jeden řádek pro každý příklad) a 4 sloupce. První sloupec pro popis systému, druhý pro popis závady, třetí sloupec pro popis chyby a čtvrtý sloupec pro popis selhání systému.

Literatura

1. Mashkov V., Fišer J. Samokontrola a samodiagnostika na systémové úrovni. *Ukrainian Academic Press*, Lviv, 2010, 176 stran.
2. Laprie, J.C., ed. Dependability: Basic concepts and terminology- in English, French, German, Italian and Japanese. *Springer-Verlag*, Vienna, Austria, 1992.
3. Laprie, J.C. Dependability of software-based critical systems*: in Dependable Network Computing*. *Kluwer Academic Publishers*, 1999.
4. Preparata F., Metze G., Chien R. On the connection assignment problem of diagnosable system. *IEEE Trans. on Electronic Computers.* Vol. EC-16, No. 12, 1967, pp. 848-854.
5. Barsi T., Grandoni T., Maestrini P. A theory of diagnosability of digital systems. *IEEE Trans. on Comp.* Vol. C-25, No. 6, 1976, pp. 585-593.
6. Vedeshenkov V. On organization of self-diagnosable digital systems. *Automation and computer engineering*. Vol. 7, 1983, pp. 133-137.
7. Fujiwara H., Kinoshita K. Some existence theorems for probabilistically diagnosable systems. *IEEE Trans. on Comp*. Vol. C-27, No. 4, 1981, pp. 297-303.
8. Mallela S., Masson G. Diagnosable systems for intermittent faults. IEEE Trans. on Comp. Vol. C-27, 1978, pp. 379-384.