Univerzita Hradec Králové
Fakulta informatiky a managementu

DISERTAČNÍ PRÁCE

2016 Monika Borkovcová

Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

ANALÝZA A OPTIMALIZACE PŘÍSTUPOVÝCH OPRÁVNĚNÍ A MODELOVÁNÍ UŽIVATELSKÝCH ROLÍ

DISERTAČNÍ PRÁCE

Autorka: Ing. Monika Borkovcová (rozená Šimková)
Školitelka: doc. RNDr. Petra Poulová, Ph.D.

Hradec Králové, září 2016

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů

Symbol	Význam
$\mathbb{N}$	množina přirozených čísel
$\cup$	sjednocení množin
$\cap$	průnik množin
$\wedge$	konjunkce, logický součin
$\vee$	disjunkce, logický součet
$\in$	operátor, je prvkem, obsažení prvku v množině
$\notin$	operátor, není prvkem, neobsažení prvku v množině
$\Rightarrow$	Implikace
$\Leftrightarrow$	Ekvivalence
$\rightarrow$	funkce z…do…
$\leftarrow$	funkce do…z…
$\emptyset, \{\ \}$	prázdná množina
$\subseteq$	je podmnožinou
$\times$	kartézský součin
$\exists$	Existuje
$\forall$	pro všechna, pro každé
$\neg$	Negace
$\lvert\ \rvert$	absolutní hodnota

Slovníček pojmů

Pojem	Význam
Adresářové služby	Adresářovou službou se rozumí specializovaná aplikace pro ukládání dat, jejich organizaci a přístup k nim. Data jsou obvykle uložena ve formě položek, přičemž každá položka obsahuje několik atributů.
Aplikační server (střední vrstva, serverová část aplikace)	Rozumí se ta část aplikace, která je umístěna v aplikační vrstvě a prostřednictvím které požaduje klient přístup k datům. ITAM komunikuje výhradně s aplikačními servery, nikdy ne s klienty.
Audit (Accounting)	Audit sleduje události, k nimž došlo v systému, zaznamenává je, popř. je také vyhodnocuje a odpovídajícím způsobem na ně reaguje.
Autentizace	Ověření identity entity (osoby)[1] nebo obecně ověření identity zdroje informací. Autentizace je vždy založena na jedné z následujících metod nebo jejich kombinaci: • Osoba něco má (např. čipovou kartu, autentizační kalkulátor apod.), • Osoba něco zná (PIN, stálé heslo, ...), • Osoba něčím je (biometrické údaje, analýza hlasu, …), Osoba něco umí udělat (např. vytvořit elektronický podpis).
Autorizace	Entita má na základě své autentizace přiřazena autorizační oprávnění.
Fixace atributu	při fixaci atributu se vygenerují role, které budou pokrývat všechny hodnoty tohoto atributu.
Identita	Totožnost. Digitální označení entity takovým způsobem, který umožňuje jeho rozlišení od jiných entit. Identita může být reprezentována různým způsobem a také více způsoby (např. Identita/Persona představující konkrétního zaměstnance), důležitá je přitom jednoznačnost v rámci stanovené oblasti působení.
Maximalizační problém (též úloha)	jedna z úloh, která se řeší při modelování rolí. Základním vstupem je zde hodnota, určující počet rolí. Systém se pak pokusí nalézt role o tomto počtu takové, aby pokrývaly co nejvíce účtů.
Minimalizační problém (též úloha)	jedna z úloh, která se řeší při modelování rolí. Základním vstupem je zde hodnota v procentech, určující kolik procent účtů se má pokrýt.

[1] V tomto výzkumu a jeho řešení je zúžen pojem entita na pojem osoba

	Systém se pak pokusí nalézt co nejmenší počet rolí, aby pokrýval zadaný počet účtů.
Org Tree	Organizational Tree, neboli organizační strom vyjadřuje hierarchii vztahu nadřízený a podřízený. Má společný kořen (TOP managera), vrstvu manažerů a koncové listy představující řadové zaměstnance.
Personální ID)	Osobní číslo zaměstnance (PersID)- jednoznačně identifikuje zaměstnance, obecně se nemění, není možné, aby bylo přiděleno jinému zaměstnanci i po jeho odchodu z organizace.
Persona (identita)	Identifikace zaměstnance v IBM Tivoli Identity Manager.
Předdefinovaná role	Role, která bude součástí následující namodelované sady rolí. Předem definované role se nastavují zaškrtnutím příslušného políčka v tabulce rolí v hlavním okně
Regulární výraz	Řetězec popisující množinu možných řetězců na základě zástupnosti za dohodnuté meta znaky.
Řízení identity (Identity Management - IM)	Strategie zahrnující procesy a postupy sloužící k identifikaci entity (např. uživatele) během jejího pracovního cyklu. Řízení identity se definuje aplikováním rolí a pravidel.
Role a pravidla	Jednotlivá oprávnění (atributy) jsou zahrnuty do organizačních rolí. Uživatel pak dostává příslušná oprávnění prostřednictvím svého přidělení k jednotlivým organizačním rolím. Pravidla pak definují zákonitosti a pravidla přidělování jednotlivých organizačních rolí.
Logonname (logon)	Přihlašovací jméno do systému, tzv. uživatelský účet.

Seznam zkratek

Zkratka	Význam
ACL	Access control list – seznam uživatelů s jejich přístupy.
AD	Active Directory - modifikované LDAP společnosti Microsoft, kam se ukládají informace o entitách v prostředí Microsoft.
COBIT	Control Objectives for Information and related Technology - neboli cílené řízení pro informační a související technologie.
CSV	Comma Separated values. Je formát textového souboru, který se sestává z řádků, ve kterých jsou jednotlivé položky odděleny čárkou.
DAC	Discretionary access control – volitelné řízení přístupu.
DSD	Dynamic Separation of Duty Relations - dynamické oddělení práv.
ISO	International Organization for Standardization – mezinárodní organizace pro standardizaci.
IT	Information Technology – informační technologie.
ICT	Information and communication technology – informační a komunikační technologie.
ITIL	IT Infrastructure Library - knihovna osvědčených postupů.
ITIM	IBM Tivoli Identity Manager – IBM Identity
JRE	Java Runtime Environment - virtuální prostředí nutné pro funkčnost aplikací naprogramovaných v jazyce Java.
LDAP	Lightweight Directory Access Protocol - přístupový protokol mezi klientem a adresářovým serverem (X. 500) na bázi TCP/IP. V současné době pod pojmem LDAP nerozumíme pouze komunikační protokol, ale i vlastní adresářový server.
LDIF	Lightweight Directory Interchange Format - standardizovaný formát, který reprezentuje data adresářového serveru jako množinu záznamů v textové formě.
MAC	Mandatory access control – povinné řízení přístupů.
NIST	National Institute of Standards and Technology - Národní institut standardů a technologií, který je součástí ministerstva obchodu vlády Spojených států amerických.
RBAC	Role Based Access Control – model řízení přístupů na základě rolí.

RTI	Research Triangle Institute – světová výzkumná instituce.
SSD	Static Separation of Duty Relations - statické oddělení práv.
UserID	Identifikace účtu zaměstnance v podřízených systémech. Příkladem UserID v ITIM je winlogonname.
TOGAF	The Open Group Architecture Framework

Prohlášení

Prohlašuji, že jsem disertační práci zpracovala samostatně a s použitím uvedené literatury a pramenů.

Hradec Králové ……..

září, 2016 Monika Borkovcová

Poděkování

Nejprve bych ráda poděkovala vedoucí disertační práce, docentce Petře Poulové za kontinuální podporu, osobní přístup a mnohdy i povzbuzení při mém Ph.D. studiu. Jistě i za možnost zapojení se do souvisejícího výzkumu a zkušenostmi, které se mnou během studia sdílela. Její vedení mi pomohlo nejen během psaní této disertační práce, ale během celého studia, jelikož mi vždy byla schopna vše vysvětlit a při tom mě i vyslechla a četně diskutovala.

Dále bych ráda poděkovala profesorovi Peteru Mikuleckému za jeho vstřícnost, rady a empatii, se kterou mi pomáhal řešit různé spletité cesty mého studia.

V neposlední řadě bych ráda uvedla poděkování mé rodině, převážně mému manželovi, a známým, kteří mě podporovali.

Abstrakt

Jednou ze zásad bezpečnosti při přístupu k informacím je dosažení co nejvhodnějšího počtu uživatelských přístupů v systémech. V rámci tohoto procesu je nutné klasifikovat všechna potřebná oprávnění přístupu, kterým zároveň předchází dostatečná analýza a zajištění odpovídající formalizace. Tento proces je označovaný jako klasifikace na základě rolí. Z hlediska pozdější správy je ideální dosažení co nejmenšího nebo nevhodnějšího počtu rolí, které jsou v souladu s potřebami organizace. Hlavním cílem při modelování uživatelských rolí je dospění ke stavu, kdy všichni uživatelé mají všechna práva, která potřebují. Současně je nutné vyvarovat se stavu, kdy některý uživatel získá větší práva, než potřebuje, v důsledku chybného tzv. katalogu rolí. Případně je výsledkem obtížně řiditelný a nesrozumitelný systém rolí. Základní část předpokládaného výzkumu se soustřeďuje na vytvoření formalizace, jež umožní provázání použitelných algoritmů a správy identit a tím jednodušší návrh a správu celého životního cyklu systému rolí. Pro základní klasifikaci je nutné provést úspěšnou implementaci a vývoj algoritmů použitelných pro analýzu přístupových oprávnění a modelování uživatelských rolí. V současné době existuje pouze výchozí klasifikace, ale bez provázanosti na použitelné algoritmy.

Klíčová slova:

Správa identit, životní cyklus rolí, RBAC, metodika, softwarové řešení správy uživatelských oprávnění.

Abstract

One of the principles of security in accessing information to achieve optimum number of user accesses in the systems. In this process, it is necessary to classify all of the necessary access privileges, which also prevents sufficient analysis and ensure adequate formalization. This process is known as classification based on roles. From the viewpoint of subsequent management is optimal to achieve the smallest number or the most appropriate number of roles, which are in accordance with the needs of the organization. The main aim in modeling user roles is to achieve result when all users have all the rights they need. It is also important to avoid inappropriete catalog of roles. At an incorrect catalog of roles, a user can have more rights than he need or is the result difficult to manage and incomprehensible system roles. The basic part of the research focuses on the creation of formalizing enabling interconnection of useful algorithms and identity management. This brings easier way to design and manage the entire lifecycle of roles. For successful implementation it is necessary, after the basic classification, development of algorithms useful for the analysis of access privileges and modeling user roles. Currently, there is only the initial classification, without linkage to the applicable algorithms.

Keywords:

Identity management, life cycle of roles, RBAC, methodology, software solution for managing user privileges.

Úvod

Systém uživatelských rolí je začleněn do oblasti bezpečnosti informačních technologií (Hu et al., 2007). Z dnešního pohledu již historickou volbou bylo řízení přístupů pomocí modelu identit, tedy manuální přidružení konkrétního účtu jedné osoby k aplikaci. Bylo to i z důvodu, že aplikace byly bez komunikace. S příchodem terminálů bylo již zapotřebí původní model rozšířit a docházelo k ověření oproti membershipům databázových systémů. S nástupem operačních systémů, podobných současným, přišel i vlastní membership. Dle (Hu et al., 2001) současné řízení uživatelských účtů vzhledem k rostoucím množstvím zdrojů a uživatelů vyžaduje cílené a kontinuální činnosti v souladu s bezpečnostní politikou organizace. Řízení uživatelských účtů je v oblasti zabezpečení informačních technologií klíčová činnost při realizaci celé informační soustavy, tedy informačního systému a ostatních napojených aplikací (Ferraiolo et al., 2007). Pro uchopení řešené problematiky je podstatné vnímat jej jako komplex celé soustavy uživatelských přístupů a oprávnění (Park, 2001). Při návrhu viz (Ray et al., 2004) a (Shin, 2000) řízení přístupů a samotném nasazení by vždy měla být splněna základní podmínka. Ta je cílena na nalezení optimálního řešení pro centralizovanou správu uživatelských účtů a rolí (Sandhu et al., 1996), což ovšem nelze aplikovat obecně a je tedy zapotřebí umět aplikovat řízení uživatelských účtů i na heterogenní prostředí. Základním principem ovšem zůstává autorizování pomocí uživatelského účtu při využití libovolného neveřejného systému IT sloužícího například pro vnitřní chod organizace nebo jen do jeho zabezpečené části, kdy dochází k přidělení přístupu ke konkrétním úlohám systémů (Mosse, 2002). Problém ovšem je v současném rozmachu informačních a komunikačních technologií (ICT), který zapříčiňuje vysokou závislost všech organizací na ICT. V článku (Doucek a Novotný, 2007) jsou uvedeny standardy pro řízení podnikové informatiky např. COBIT, ISO / IEC 9000 skupinových a ITIL, kde autoři zmiňují ovlivňování většiny původních procesů při inovacích ICT. (Feltus, Dubois a Petit, 2010) ve svém článku navrhují způsob řešení uživatelských rolí (RBAC) v souladu s požadavky COBITu s využitím přenosového bodu z COBITu do RBAC. Tento přenosový bod je v podstatě kopírování odpovědnosti daného zaměstnance z COBITu do RBAC, což ovšem často neodpovídá skutečnosti, jelikož není vždy odpovědný za všechny aktivity, které má ve své náplni práce nebo naopak. Rozvrhování uživatelských rolí musí na prvním místě vždy splňovat dle (Hu et al., 2008) odpovídající ochranu informací a systémů před

poškozením nebo zneužitím a na straně druhé (Hu et al., December, 2008) řádnou úroveň přístupových oprávnění do informační soustavy. Tento proces v sobě skrývá nejen prvotní, statickou implementaci množiny uživatelských rolí, ale udržitelnost katalogu rolí v rámci celého životního cyklu uživatele. Převážně nalezení takového východiska, které v sobě zahrnuje každodenní, rutinní, vstupy či výstupy uživatelů do/z systému. Prvotním cílem práce je analyzovat několik vybraných systémů v komerční i státní sféře a nalézt společné vzory chování v životním cyklu uživatele. Na základě získaných dat formalizovat jednotlivé vstupní parametry a ověřit získané poznatky. Pro ověření předkladů vytvořit aplikaci, která umožňuje generovat výstupní sadu přístupových rolí dle předem zadaných kritérií s využitím modelu řízení na základě rolí tzv. Role Based Access Control (RBAC).

Východiskem práce je rozsáhlá případová studie, kdy informace o uživatelských účtech (dále data) byla sbírána a postupně analyzována při reálně se měnících podmínkách a i organizačních strukturách, při zavádění nových informačních systémů nebo při inovacích ze široké škály strategických cílů organizací, převážně z důvodu zvýšení konkurenceschopnosti. Případové studie probíhaly i v rámci projektu 5.1 SPK01/013 „Hradecký IT klastr“. Poznatky z těchto analýz přinesly příležitost zkoumat bezpečnostní politiku organizací v oblasti uživatelských přístupů s cílem praktického nalezení vhodných vzorů chování uživatelských rolí vzhledem k jejich implementaci a vypozorovat takové změny, které lze zapojit do strategického plánování organizací v souladu s její organizační strukturou.

Struktura disertační práce

Kromě úvodu se tato práce skládá z *pěti* hlavních částí:

- ✓ *První* úsek popisuje obecný úvod do současného vědeckého řešení tématu včetně základních teoretických východisek, generální formalizace modelování uživatelských rolí a určení souvisejících otázek.
- ✓ *Druhá* část práce prezentuje hlavní výsledky výzkumu v rámci projektu 5.1 SPK01/013 „Hradecký IT klastr“, na kterém se autorka po dobu tří let podílela.
- ✓ Jako *třetí* díl práce následuje provázání vybraných návrhových vzorů RBAC s případovými studiemi včetně základního popisu provozních informačních systémů. Cílem je nalezení doporučení nasazení RBAC pro různé případové studie. Tato kapitola řeší i problematiku skutečné pracovní náplně uživatele a jeho aktivit v systému.
- ✓ Předposlední a *čtvrtý* oddíl popisuje problematiku vývoje a inovací v organizacích a dynamiku uživatelských rolí včetně implementace tzv. vektoru změn.
- ✓ Závěrečná *pátá* část práce shrnuje výsledky tohoto výzkumu a nastiňuje možné směry dalšího vývoje.

Cíle práce

Hlavním cílem této práce je nalezení společných vzorů chování životního cyklu uživatelů a uživatelských přístupů pro různé případové studie s využitím rozšířené formalizace provázané na vhodné algoritmy.

Pro naplnění hlavního cíle je nutné dosáhnout následujících nezbytných dílčích cílů:

a) Analyzovat nastavení přístupových oprávnění a vzorů chování uživatelských rolí a provést výběr vhodných algoritmů pro tvorbu testovací aplikace.
b) Formalizovat vstupy získané z předešlé analýzy a provázat je na vybrané algoritmy.
c) Realizovat testovací aplikaci jako nástroje pro generování výstupní sady rolí.
d) Popsat RBAC návrhové vzory a na základě výstupních katalogů uživatelských rolí nalézt propojení s návrhovými vzory a tím vytvořit doporučení nasazení RBAC pro různé případové studie.
e) Identifikovat zranitelná místa pro problematiku skutečné pracovní náplně uživatele a aktivit v systému.
f) Nalézt doporučení pro udržitelnost správného katalogu rolí z pohledu životního cyklu uživatele, vývoje a inovací v organizaci.

1 Teoretický základ a literární rešerše

V současné době je v každé moderní organizaci cílem ochraňovat informace, naplnění tohoto cíle umožňuje informační bezpečnost obecně, jejichž součástí je i řízení přístupů.

Pro přehlednost jsou zde uvedeny mezinárodní normy Mezinárodní organizace pro normalizaci (ISO[2]) spadající pod Informační technologie – Bezpečnostní techniky (ISO, 2016):

- ISO/IEC 27000 Systém řízení bezpečnosti informací definující základní normy a terminologie.
- ISO/IEC 27001 Systém řízení bezpečnosti informací, jedná se o hlavní normu, dle níž probíhá certifikace.
- ISO/IEC 27002 Soubor postupů pro opatření bezpečnosti informací.
- ISO/IEC 27003 Směrnice pro implementaci systému řízení bezpečnosti informací.
- ISO/IEC 27004 Řízení bezpečnosti informací – měření.
- ISO/IEC 27005 Řízení rizik bezpečnosti informací.
- ISO/IEC 27006 Požadavky na orgány poskytující audit a certifikaci systémů řízení bezpečnosti informací.
- ISO/IEC 27007 Směrnice pro audit systémů řízení bezpečnosti informací
- ISO/IEC TR 27008 Směrnice pro audit opatření *Systému řízení bezpečnosti informací* (ISMS[3]).
- ISO/IEC 27010 Směrnice pro řízení bezpečnosti informací pro meziodvětvové komunikace a komunikace mezi organizacemi.
- ISO/IEC 27011 Směrnice pro řízení bezpečnosti informací pro telekomunikační organizace na základě ISO/IEC 27002.
- ISO/IEC 27013 Návod pro integrovanou implementaci ISO/IEC 27001 a ISO/IEC 20000-1.
- ISO/IEC 27014 Správa bezpečnosti informací.
- ISO/IEC TR 27015 Směrnice pro řízení bezpečnosti informací pro finanční služby.

[2] International Organization for Standardization
[3] Information Security Management System

- ISO/IEC TR 27016 Řízení bezpečnosti informací – Organizační ekonomika.
- ISO/IEC 27017 Kodex pro řízení bezpečnosti informací pro cloudové služby.
- ISO/IEC 27018 Kodex pro ochranu osobních informací ve veřejných cloudech (PII procesory).
- ISO/IEC 27019 Řízení bezpečnosti informací pro energetický průmysl, založený na ISO/IEC 27002.
- Informační strategie

1.1 Přístupy k řízení informačních technologií

Tato kapitola přináší základní ovšem nutný nikoliv úplný přehled k přístupům řízení informačních technologií. Obecně z důvodu návaznosti na samostatné řízení přístupů do informačních systémů v organizaci.

1.1.1 Metodika ITIL

ITIL neboli IT Infrastructure Library je knihovna osvědčených postupů[4] poskytující organizaci způsoby, doporučení, rady, ale i popisující hrozby v oblasti řízení informačních technologií, jež jsou nazývány jako IT služby, tedy na řízení IT je pohlíženo jako na službu. Oproti výše vypsaným normám se nejedná o standardy, ovšem v doporučení z nich ITIL vychází, ale již bez nutnosti striktního dodržování. ITIL v sobě zahrnuje strategii služeb, návrh služby, přechod služeb, provoz služeb a soustavné zlepšování služeb a další doplňkové knihy či oblasti v rámci životního cyklu IT služby. (Brunnstein, 2006; Dressler et al., 2010)

Hodnocení aplikace těchto metodik do praxe se již v prvopočátcích intenzivně zabýval Hochstein např. v (Hochstein et al., 2005), kde prováděl zkoumání ITIL na čtyřech případových studiích, ve všech uvedl, že uplatnění ITIL je výhodné z hlediska snižování nákladů. Ve vztahu k řízení bezpečnosti a aplikace ITILu lze uvést Sheikhpour a Modiri (Sheikhpour a Modiri, 2012), kteří kriticky, jak samotní autoři uvádějí, hodnotí důležitost efektivního řízení informační bezpečnosti ve spojení s ISO/IEC 27001 a způsob, jak jej sladit s ITIL. Dle (Yue a Jian, 2011) je přínosem ITIL zefektivnění vnitřních procesů dané organizace, ovšem často nedochází k úplnému dodržení všech doporučení. Toto souvisí i

[4] Také označovaných jako best practices.

se samotnou problematikou nastavení přístupových oprávnění, která by měla ideálně odpovídat pracovnímu zařazení daného pracovníka.

1.1.2 Metodika TOGAF

Oproti ITIL je podstatou TOGAF[5] vytvoření enterprise architektury, která má být nápomocná k tvorbě celé IT architektury ve smyslu infrastruktury. Hlavním cílem je sjednocení všech procesů do integrovaného prostředí a tím zajistit rychlejší reakci na potřebné změny včetně řešení bezpečnosti. Metoda vývoje architektury (ADM[6]) se dělí na čtyři klíčové oblasti, a to architektura podniku, dat, aplikací a technologií. Každá oblast se soustředí na význam dle svého názvu. Architektura podniku zahrnuje podnikovou strategii, klíčové procesy, organizaci podniku a samotný způsob řízení. Architektura z pohledu dat se soustředí na datové zdroje a aktiva využívaná v organizaci. Architektura aplikací přináší jakousi šablonu k zapojení jednotlivých aplikací do podnikových procesů a řeší i jejich vzájemnou komunikaci a poslední architektura technologií se zabývá hardwarovými a softwarovými prostředky a nástroji. Při aplikaci metodiky TOGAF je nutné dodržet předem definované fáze. (Buckl et al., 2009) Cabrera et al. zdůrazňuje přínos TOGAF, i přes občas přílišnou abstrakci, a to zlepšení stávajících podnikových procesů díky komplexnímu a koordinovanému způsobu pro dosažení cílů organizace, které odpovídají podnikové strategii. (Cabrera et al., 2016)

1.1.3 Metodika COBIT

COBIT[7] neboli cílené řízení pro informační a související technologie lze pokládat za rámec, který se snaží maximálně využít dostupné zdrojem a minimalizovat IT rizika, součástí této strategie je propojení řízení podniku a řízení a správu informatiky. Podstatnou součástí jsou kompetence a odpovědnost jednotlivých vlastníků podnikových a IT procesů. Současný stav nasazení COBIT v mezinárodním měřítku ve vztahu k velikosti organizace a i tvorbou hodnoty podniku řeší De Haes a kolektiv. (De Haes et al., 2016)

5 The Open Group Architecture Framework
6 Architecture Development Method
7 Control Objectives for Information and related Technology

1.2 Základní pravidla bezpečnosti

Dle (Ahn et al., 2000) ochrana informací v počítačových systémech s víceuživatelskými přístupy vyžaduje obecné zavádění při vývoji tak, aby bylo co nejrychlejší, efektivní a ekonomicky méně náročné.

Pro zachování základních pravidel bezpečnosti je třeba zachovat tři základní ochranná opatření, a to prevenci, detekci (tedy odhalení neautorizovaného využití informací) a obnovu (Gollmann, 2010).

Dle normy ISO/IEC 27000 jsou dále vyjmenované vybrané pojmy související s tématem (ISO, 2016):

- Autenticita vyjadřuje, že entita je tím, za co se vydává.
- Autentizace je poskytnutí záruky, že prohlašovaná charakteristika entity je správná.
- Dostupnost je přístupnost a použitelnost na žádost oprávněné entity.
- Důvěrnost odpovídá splnění požadavku.
- Cíl je výsledek, kterého má být dosaženo.
- Hrozba popisuje potencionální příčinu nechtěného incidentu, jehož výsledkem může být poškození systému nebo organizace.
- Informační systém vystihuje aplikace, služby, aktiva informační technologie nebo další komponenty zacházející s informacemi.
- Integrita prezentuje přesnost a úplnost.
- Kompetence poukazuje na schopnost použít znalosti a dovednosti k dosažení cílených výsledků.
- Opatření, řízení, kontrola odpovídá souhrnně prostředkům modifikující riziko.
- Organizace představuje osobu nebo skupinu osob, které mají své vlastní funkce s odpovědnostmi, pravomocemi a vztahy, pomocí nichž mohou dosáhnout svých cílů.
- Politika cílí na celkový záměr a směřování organizace.
- Požadavek lze popsat jako potřebu nebo očekávání, které jsou stanovené, obecně předpokládané nebo závazné.

- Proces řeší soubor aktivit majících vzájemný vztah nebo vzájemně na sebe působí a přeměňují vstupy na výstupy.
- Riziko ztvárňuje dopad nejistoty na dosažení cílů. Riziko bezpečnosti informací vystihuje spojení s možností, že hrozby využijí zranitelného místa (zranitelnost) informačního aktiva nebo skupiny informačních aktiv a tím způsobí organizaci škodu.
- Řízení přístupu se vyznačuje procesem zajišťujícím, aby přístup k aktivům byl autorizován a omezen na základě obchodních pravidel a bezpečnostní politiky organizace.
- Událost rovná se výskytu nebo změně určité množiny okolností.
- Útok představuje pokus o zničení, vystavení hrozbě, změnu, vyřazení z činnosti, zcizení aktiva nebo získání neoprávněného přístupu k aktivu nebo uskutečnění neoprávněného použití aktiva.
- Zranitelnost znamená slabé místo aktiva nebo opatření, které může být využito jednou nebo více hrozbami.

K řízení přístupu se k vynucení zabezpečení a eliminaci informačních bezpečnostních rizik spojených s uživatelským přístupem do informačních systémů používají dle (Alturi a Ferraiolo, 2011) tři základní principy známé pod zkratkou *CIA*:

- Utajení (Confidentiality) - odkazuje na nutnost udržet informace v bezpečí a soukromí.
- Integrita (Integrity) - poukazuje na koncept ochrany informace od její nežádoucí změny po změnu neoprávněnými uživateli. Jedná se o ochranu informací před záměnou či modifikací neautorizovaným uživatelem při zachování pravidla, že citlivé údaje nemůže nikdo změnit než uživatel sám nebo správce daného systému.
- Dostupnost (Availability) - řeší stálou dostupnost informace v případě potřeby uživatele.

1.3 Typy řízení přístupů

Kritéria hodnocení spolehlivosti počítačových systémů[8], definuje dva základní typy řízení přístupů, a to volitelné řízení přístupu a povinné řízení přístupu.

[8] The Trusted Computer System Evaluation Criteria (TCSEC), norma ministerstva obrany vlády Spojených států amerických, označováno jako „Oranžová kniha".

Volitelné řízení přístupu (DAC)[9] je založeno na seznamu pro řízení přístupu (ACL)[10]. Tedy na totožnosti uživatele, což v důsledku znamená, že každý zdroj má svého vlastníka. Vlastník je tak schopný kontrolovat, kdo přistupuje ke zdrojům, které vlastní. Tento typ přístupu je často používán v operačních systémech. Operační systém totiž často obsahuje řízení přístupu na základě ACL.

Oproti tomu povinné řízení přístupu (MAC)[11] má za cíl omezovat uživatele se zdroji na základě bezpečnostních atributů spojených s uživateli a zdroji. Na rozdíl od volitelného přístupu je povinné řízení řízeno vyšší autoritou (Latham, 1986).

Moderní informační systémy tvoří obrovské množství zdrojů a uživatelů. Složitost těchto systémů těžko umožňuje aplikovat volitelné řízení přístupů a na druhou stranu povinné řízení přístupu je příliš omezující obecnou strukturou informačních systémů (Ferraiolo et al., 2006, Hu et al., 2011). Role-based access control (RBAC) se ukázala jako hlavní alternativa k volitelnému a povinnému řízení přístupů (Ferraiolo a Kuhn, 1992, Sandhu et al., 1996).

1.4 Vymezení základních pojmů spojených s RBAC

V předešlé kapitole byla zmíněna základní pravidla bezpečnosti spojená s modelováním uživatelských rolí a typy řízení přístupů. Vzhledem k používaným termínům v následujících kapitolách a stále se opakujícího názvosloví v RBAC je zapotřebí nejdříve formálně tyto termíny popsat. Dle (Ferraiolo et al., 2006):

- *Membership* = vnitřní technická realizace požadované autorizace[12].
- *Uživatelský účet* = jedná se o sadu atributů s patřičným ID. Toto ID může být simultánně aktivní s jiným.
- *Uživatel (USERS)*= konkrétní osoba využívající počítačový systém, který může mít i několik uživatelských účtů.
- *Role (ROLES)* = jde o pravomoc a odpovědnost svěřenou uživateli. Ideálně by měla odpovídat pracovnímu zařazení v kontextu celé organizace, což ovšem je často odlišné od reality.

[9] Discretionary access control (DAC) – volitelné řízení přístupu.
[10] Access control list – seznam uživatelů s jejich přístupy.
[11] Mandatory access control (MAC) – povinné řízení přístupů.
[12] Autorizace – entita má na základě své autentizace přiřazena autorizační oprávnění. Autentizace je ověření identity entity (v rámci řešení výzkumného cíle chápáno jako osoba) nebo obecně ověření identity zdroje informací.

- *Sezení (session, user_sessions, session_roles)* = stav po autorizaci a zpracování membershipu, který trvá po celou dobu platné autorizace.
- *Subjekt (S)*= aktivní entita jako je člověk, počítačový proces vykonávaný jménem uživatele, uživatel může mít několik běžících subjektů zároveň za podmínky jednoho id a tedy jedné session.
- *Operace (OPS)*= označovaná i jako transakce, je aktivní proces vyvolaný subjektem.
- *Objekt (OBS)* = jakýkoliv zdroj v počítačovém systému je definován jako objekt a jedná se o pasivní entitu, která obsahuje nebo získává informace.
- *Přístup* = interakce mezi subjektem (S) a objektem (OBS).
- *Řízení přístupu* = proces řízení přístupu k objektům (OBS) – přístup je povolen k autorizovaným subjektům (S).
- *Oprávnění (PRMS)* = autorizace k vykonání samotné akce v počítačovém systému, daná operace použitá v různých objektech (OBS) představuje rozdílná oprávnění a analogicky různé operace aplikované na jeden objekt představují rozdílná oprávnění, interakce subjektu (S) s objektem (OBS).
- *Přiřazení oprávnění rolím (PA – Permission Assignment)* = navázaná relace mezi rolemi (ROLES) a oprávněními (PRMS).
- *Přiřazení rolí uživatelům (UA – User Assignment)* = relace mezi uživateli (USERS)a množinou rolí (ROLES)
- *Hierarchie rolí (Role Hierarchy)* = uspořádaná hierarchie rolí.

A dále speciální vztahy mezi těmito pojmy, které jsou podrobně zmíněny již v následujících kapitolách.

1.5 Vývoj RBAC

V roce 1992 (Ferraiolo a Kuhn, 1992) představil David Ferraiolo (NIST)[13] a Richard Kuhn (NIST) pod zkratkou RBAC, řízení přístupů pomocí rolí s využitím hierarchie a rozsáhlými omezeními pro organizování přístupů. Hlavním důvodem vzniku byla možnost komerčního využití, které doposud Mandatory Access Controls (MAC) neposkytovalo. Povinné řízení přístupu bylo využíváno převážně pro vojenské účely a toto řízení přístupů v podstatě nemohlo respektovat potřeby komerčních organizací. RBAC se stal

[13] Národní institut standardů a technologií, který je součástí ministerstva obchodu vlády Spojených států amerických.

převládajícím a využívaným modelem, o čemž svědčí i prestižní jména některých komerčních společností, které začaly od roku 1994 na základě modelu RBAC stavět vlastní produkty. Model RBAC byl v roce 2000 spojen s modelem RBAC od (Sandu et al., 1996), jehož představiteli jsou Ravi Sandhu, Edward Coyne, Hal Feinstein a Charles Youman. Tento model hlouběji a systematicky popsal referenční modely, které přispěly k řešení různých komponent RBAC a jejich vzájemné interakci. V roce 2000 došlo spojením k vytvoření jednotného modelu pro RBAC (Sandhu et al., 2000), který je označen jako NIST RBAC (INCITS, 2004), který byl v roce 2004 přijat jako standard ANSI/INCITS (INCITS 359-2004). Tento standard obsahuje referenční model a funkční specifikaci pro typické znaky RBAC definované v referenčním modelu (Ferraiolo et al., 2007). V roce 2010 došlo k retrospektivní analýze ekonomických dopadů při nasazení RBAC (O'Connor a Loomis, 2010). Organizace Research Triangle Institutu (RTI)[14] zjistila, že model RBAC využívají organizace mající 500 a více uživatelů. Tato zajímavá analýza z ekonomického pohledu zdůrazňuje současné hojné využití sdílených zdrojů, ale i s tím spojené nebezpečí úmyslného či neúmyslného zneužití či modifikace systémů. Zmiňuje nutnost častých a rychlých změn podnikových procesů, vzhledem ke stále se zvyšujícím požadavkům na inovativnost, legislativu a technologické prostředky. Dle (Ferraiolo et al., 2011) tím dochází k ohrožení integrity, důvěrnosti a dostupnosti informací organizace a tudíž i její infrastruktury. Poukazuje na výhody RBAC, od nejzákladnější přednosti nahrazení identity uživatele rolí (utajení)[15] po úsporu času na správu systému a s tím i snížení celkových nákladů organizace.

Díky možnosti změny práv všem uživatelům přiřazeným v konkrétní roli dochází k omezení chybovosti lidského faktoru. (Hu et al., 1993) popisuje i zásadní nutnost, a to podobnost organizační struktury v instituci a nasazeného systému uživatelských rolí. Richard Kuhn v roce 2010 představuje další generaci standardu INCITS 359-2004 (Kuhn et al., 2010) a v roce 2012 došlo k aktualizaci standardu na INCITS 359-2012 (INCITS, 2012) ve spolupráci s Timem Weilem z University of Denver. Tato aktualizace standardu v sobě zahrnuje podporu atributů, bezpečnostní politiku v reálném čase a shodu s RIIS (ANSI 459-2011). Vychází tedy z původního standardu INCIST 359-2004 a zároveň

[14] Research Triangle Institute (RTI) je jedna z předních světových výzkumných institutů, původně založena za účelem zkvalitnění výzkumu třech univerzit – North Carolina State University, University of North Carolina a Duke University.

[15] Řízení identity (Identity management) – strategie zahrnující procesy a postupy sloužící k identifikaci entity během jejího pracovního cyklu. Řízení identity se definuje aplikované rolí a pravidel.

poskytuje mechanismus pro rozhodnutí začlenění atributů do řízení přístupů. Standard RIIS (ANSI 459-2011) s názvem RBAC implementace a interoperabilita je norma usnadňující interoperabilitu RBAC modelu s cílem dodržení konzistence sad komponent RBAC, což umožňuje přenesení z jednoho RBAC do druhého. Pro úplnost je nutné zmínit i standard RPE – 494-2011, který rozšiřuje RBAC o dynamický model použitím omezujícího modelu RBAC. Poskytuje funkcionální specifikaci k udržení vztahu mezi rolemi a dynamickým omezením práv, definuje rozsah a kontext pro vztahy role-role, uživatel-role a atributy citlivé na dynamické omezení práv, které mohou být implementovány v běhovém prostředí.

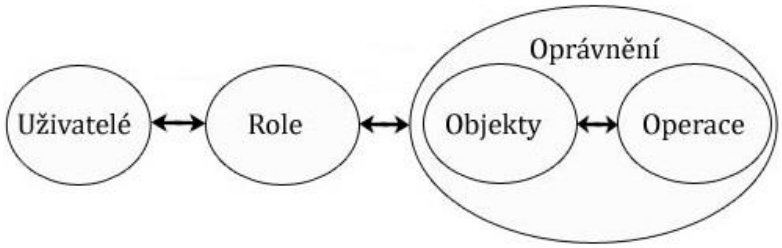
1.6 Základní popis RBAC

V návaznosti na předešlé kapitoly lze popsat samotný RBAC. Nejdříve je ale stěžejní uvést důležitý princip při modelování uživatelských rolí. Jedná se o tzv. postup selektivního přiřazování povolení uživatelům označovaný jako nejnižší privilegium. Jeho cílem je zaručit, aby uživatel nemohl provádět zbytečné a potenciálně škodlivé činnosti jako vedlejší účinek udělení oprávnění vykonávat požadované funkce (Ferraiolo et al., 2006). Samotný RBAC řeší přístupová oprávnění přidělovaná pomocí rolí. Oproti základní definici role, dle (Barkley et al., 1997, Ferraiolo et al., 1999, Portela, 2010) lze roli chápat jako kombinaci, která udává definice přístupů, tedy uživatelských účtů. V tomto pojetí je uživatelský účet nejčastěji tvořen atributy, které obsahují přístupové informace, specifická technická nastavení a personální identifikace daného uživatele. To vše lze hromadně nazvat atributy (Al-Kahtani et al., 2002). (Coyne et al., 2011) popisují oprávnění v systému rolí, kdy jsou sdružována do komplexních souhrnů. Komplexní souhrn definuje jako úlohy, které daný uživatel vykonává v rámci svého zařazení. Uživateli je tak přidělena role obsahující komplex oprávnění, nezískává tedy jednotlivé oprávnění. V případě uživatelů mající shodná pracovní zařazení je možné definovat a hromadně spravovat celé skupiny uživatelů vlastnících stejná oprávnění.

RBAC má význam v organizacích s větším počtem uživatelů, v jiném případě je vhodné využití přístupu přes ACL (Hu et al., 2006). U většího počtu uživatelů je identita[16] a přístupový management složitější a to vyžaduje rozsáhlejší bezpečnostní politiku a

[16] Identita = Totožnost. Digitální označení entity takovým způsobem, který umožňuje jeho rozlišení od jiných entit. Identita může být reprezentována různým způsobem a také více způsoby (např. Identita/Persona představující konkrétního zaměstnance), důležitá je přitom jednoznačnost v rámci stanovené oblasti působení.

přístup obecně. Rozvíjí základní principy MAC, kdy přejímá základní pravidlo, při kterém má uživatel minimální možné množství oprávnění (Ni et al., 2009). Dle (Flanagin et al., 2007) RBAC je pravděpodobně nejvýznamnější inovací ve správě identit a přístupů. Dle (Sandhu et al., 2000) snižuje administrativní zátěž díky hierarchii rolí. Hierarchie rolí snižuje počet nastavení oprávnění ve vztahu role-oprávnění a zvyšuje efektivnost ve vztahu role-uživatel, ale dle (Hu et al., 2006, June) přináší i složitější řízení a správu přístupů.

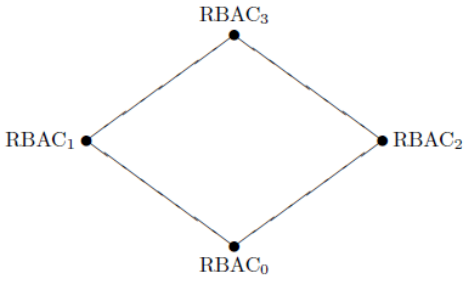


Obrázek 1 - Model řízení přístupů na základě rolí (Sandhu et al. 1996)

Obrázek 1 ukazuje mezivrstvu, která tvoří skupinu rolí vznikající při přidělování přístupových práv. Jedná se o spojení mezi uživatelem a jeho oprávněními. Role a oprávnění jsou spojeny aktivitami, které by měly být v souladu s pracovní náplní uživatelů. Uživatel své oprávnění získá aktivací role, která uživateli dané oprávnění přidělí. I uživatelé tvoří své skupiny, tedy skupiny uživatelů, jelikož na jednom pracovišti může více uživatelů vykonávat stejnou činnost. Množina „Uživatelé“ uvedená na obrázku 1 představuje vrstvu skládající se ze samotných uživatelů s rozdělením na vlastní skupiny (Ferraiolo et al., 1999, Ahn a Hu, 2007).

1.7 Klasifikace modelů RBAC96

Model RBAC z roku 1996, který byl základem pro vytvoření standardu ANSI/NIST RBAC obsahuje 4 základní modely, jejichž vzájemné vztahy jsou uvedeny na obrázku 2, je kategorizován na základní RBAC, hierarchické RBAC, omezující RBAC a symetrický model RBAC. (Moon et al., 2004)

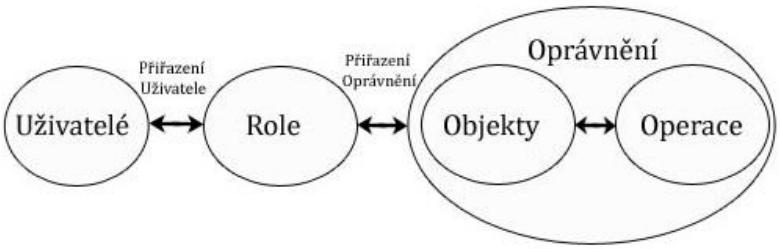


Obrázek 2 - Vztahy mezi jednotlivými modely RBAC z roku 1996

Všechny modely předpokládají systém s podporou RBAC. Základní model $RBAC_0$ vyžaduje nejnižší požadavky na systém, z tohoto důvodu je i na obrázku 2 umístěn dole. Modely $RBAC_1$ a $RBAC_2$ musí splňovat $RBAC_0$ a zároveň plní funkci nezávislosti. $RBAC_1$ uplatňuje hierarchii rolí tedy dědičnost oprávnění a $RBAC_2$ zavádí omezení na povolené konfigurace různých komponent RBAC. $RBAC_1$ a $RBAC_2$ jsou navzájem nezaměnitelné. Konsolidovaný model $RBAC_3$ zahrnuje $RBAC_1$ a $RBAC_2$ a díky tranzitivitě i $RBAC_0$. (Sandhu et al. 1996) Pro lepší pochopení dalšího výkladu jednotlivých modelů RBAC je nutné zmínit skutečnost, že jádro systému RBAC podporuje mnohočetné vztahy mezi jednotlivými datovými elementy (oboustranná šipka v obrázcích) (Ferraiolo, 2001).

1.7.1 $RBAC_0$ – základní model

(Ferraiolo, et al., 2001) uvádí, že základní model definuje pět základních datových elementů, a to uživatel, role, objekt, operace, oprávnění a vzhledem k tomu, že dochází ke spojení při správné autorizaci uživatele, tak i sezení (session). Při sezení dochází k mapování mezi uživatelem a množinou či podmnožinou rolí, která jsou uživateli přiřazena



Obrázek 3 Základní model RBAC - $RBAC_0$ (INCITS 2004, Sandhu et al. 2000)

Obrázek 3 zobrazuje přiřazení uživatele a přiřazení oprávnění. Oprávnění jsou v primární fázi množina použitých symbolů k popisu operací.

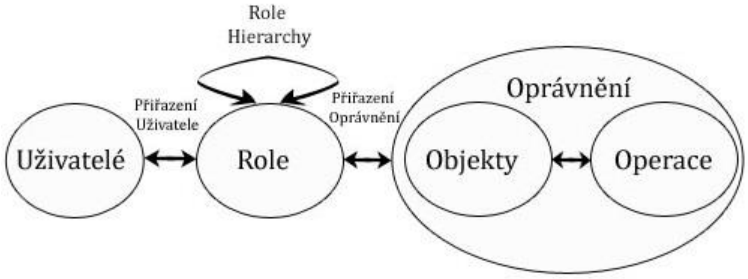
Session je tak pod kontrolou jednotlivých uživatelů, kdy uživatel může vytvořit a vybrat proces některých podmnožin rolí. Role se tak aktivuje[17] v okamžiku navázání spojení a analogicky je stejným způsobem ukončena.

Uživatel může vystupovat v jedné či více rolích, role má přiřazenou informaci týkající se autority a odpovědnosti v pracovní funkci, oprávnění přiřazená roli mohou být přístupová práva a systémová práva, oprávnění určují, jaké operace lze přiřazením dané role vykonávat nad objekty. (Sandhu et al., 1996)

1.7.2 $RBAC_1$ – hierarchický model

$RBAC_1$ rozšiřuje $RBAC_0$ o hierarchii uživatelských rolí. Rozlišuje tzv. *senior a junior role*, kdy senior role získává oprávnění od svých junior rolí a junior role dědí uživatele ze svých seniorů, toto se může lišit, záleží na implementaci (Bertino, 2003). Při uplatnění hierarchického modelu platí následující:

- určuje hranice mezi autoritou a odpovědností (Sandhu et al., 1996),
- role s většími oprávněními je přirozeně nadřazená (směrem nahoru) (Sandhu et al., 1996),
- role s menšími oprávněními je přirozeně podřazená (směrem dolů) (Sandhu et al., 1996),
- dědění autority je tak od shora dolů, (Ferraiolo a Gavrila, 2009)
- dědění odpovědnosti rolí od spodu nahoru (Ferraiolo a Gavrila, 2009).



Obrázek 4 Hierarchický model RBAC - $RBAC_1$ (INCITS 2004, 2012, Sandhu et al. 2000)

$RBAC_1$ pracuje s obecnou nebo různě omezenou (limitovanou) hierarchií rolí. Obecná hierarchie rolí umožňuje libovolné uspořádání včetně několikanásobného uplatnění

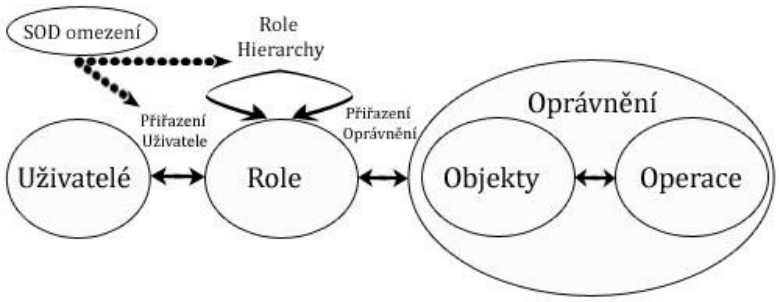
[17] tzn. je přiřazena uživateli po jeho autorizaci.

principu dědičnosti. Limitovaná hierarchie rolí uplatňuje nějaká omezení, např. stromové grafy.

1.7.3 $RBAC_2$ – omezující model

Model $RBAC_2$ přidává možnost rozdělení práv mezi několik uživatelských rolí (SOD[18]), tedy oddělení úloh jednotlivých uživatelů. Používá se v souladu s bezpečnostní politikou organizace z důvodu, aby nedošlo ke střetu zájmů a k zabránění v překročení přiměřené úrovně pravomocí pro danou pracovní pozici (Yu a Brewster, 2012). Jinými slovy dodržení patřičného stupně autority. Rozlišuje se tzv. statické a dynamické oddělení práv nebo také statické a dynamické omezení práv. (Sandhu et al., 2000)

Statické oddělení práv (SSD)[19] definuje vzájemně disjunktní přiřazení v souladu se sadou rolí. RBAC je implementováno tak, aby oprávnění jednoho uživatele nebylo v konfliktu se sadou rolí, z čehož vyplývá, že je striktní (Obrázek 7). Při uplatnění SSD je zapotřebí sledovat hierarchii rolí, jelikož SSD lze převzít z předchůdce, tím by došlo k porušení principu v souladu se sadou rolí. (Ferraiolo, 2004)

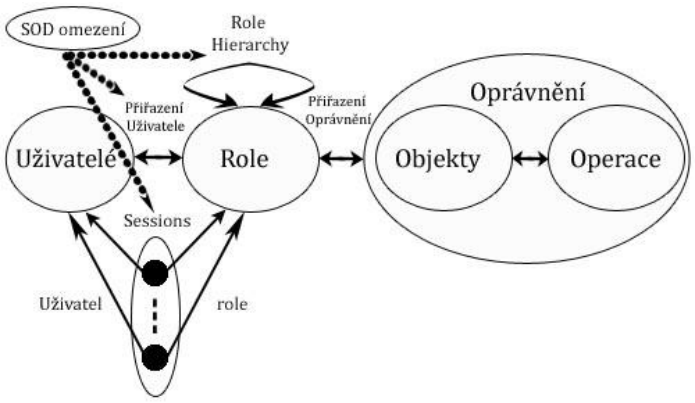


Obrázek 5 Omezující model RBAC – $RBAC_2$ statické oddělení práv (INCITS 2004, 2012, Sandhu et al. 2000)

Dynamické oddělení práv (DSD[20]) oproti SSD umožňuje uživateli být ve více rolích, které jsou konfliktní. Dynamické oddělení práv neboli dynamické oddělení úloh má schopnost adresovat potenciální střet zájmů, pomocí session zajišťuje, aby konfliktní role neběžely současně. V případě, že jsou provedeny nezávisle, pak není zapotřebí stanovovat dodatečná pravidla, v případě aktivních rolí souběžně tedy závisle, jsou vytvářena speciální pravidla. Na Obrázku 8 může být uživatel v roli např. Pokladník i v roli Vedoucí pokladník, kde Vedoucí pokladník je oprávněn odsouhlasit změny pohybu hotovosti dle role Pokladník. V situaci, kdy uživatel v roli Pokladník vypne či přepne roli Vedoucí

[18] SOD – zkratka z anglického názvu Separation of Duty Realitons
[19] SSD – zkratka z anglického názvu Static Separation of Duty Relations
[20] DSD – zkratka z anglického názvu Dynamic Separation of Duty Relations

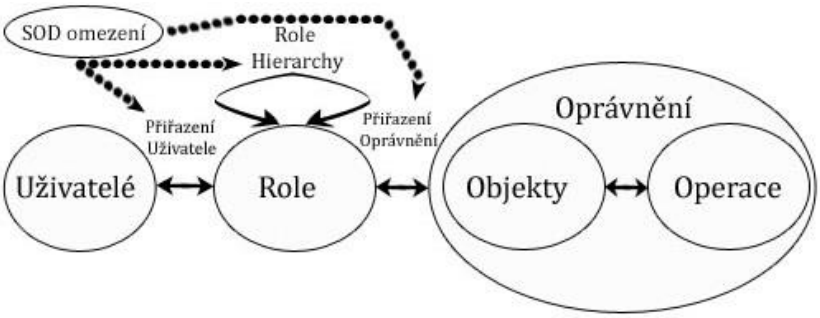
pokladny, tak v tento okamžik RBAC bude požadovat zastavení role od uživatele v roli Pokladník. Hierarchie rolí má v DSD oproti SSD omezenou vazbu, což má za následek, že autorizace a aktivace mohou být vnímány jako dva nezávislé pojmy. (Ferraiolo et al., 2003)



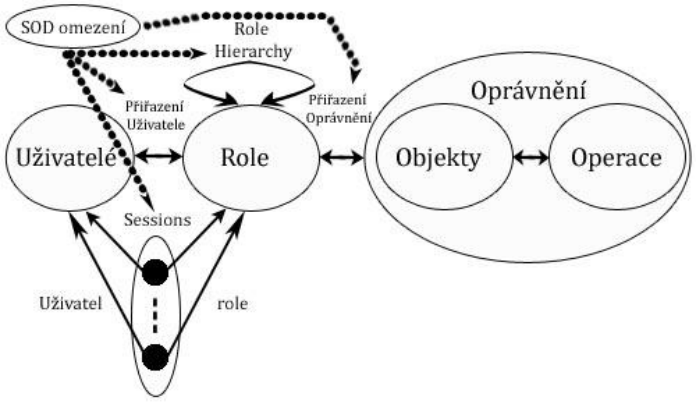
Obrázek 6 Omezující model RBAC – RBAC₂ dynamické oddělení práv (INCITS 2004, 2012, Sandhu et al. 2000)

1.7.4 RBAC₃ – symetrický model

RBAC3 kombinuje přidání omezení práv a hierarchie uživatelských rolí resp. je kontrolováno přiřazení oprávnění. Významnou roli zde má zachování nejnižšího privilegia. Podstatou je udržení přesného přiřazení oprávnění souvisejícími systémovými objekty, které korespondují s uživatelovými autorizovanými funkcemi nebo povinnostmi v rámci instituce. (Moon et al., 2004)



Obrázek 7 Symetrický model RBAC – RBAC₃ statické oddělení práv (INCITS 2004, 2012, Sandhu et al. 2000)



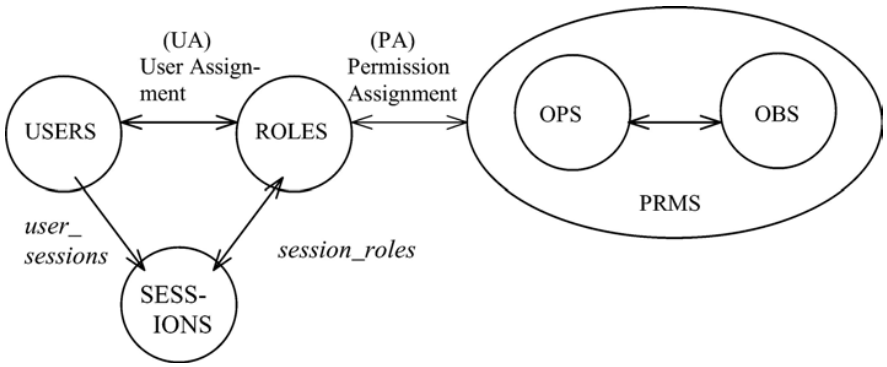
Obrázek 8 Symetrický model RBAC – RBAC$_3$ dynamické oddělení práv (INCITS 2004, 2012, Sandhu et al. 2000)

1.8 ANSI-RBAC

Americký národní institut standardů (ANSI) přijal standard pro RBAC, který poskytuje konzistentní a jednotnou definici RBAC. ANSI-RBAC standard rozlišuje tři komponenty, a to základní jádro RBAC, hierarchické RBAC a omezující RBAC. (INCITS, 2004)

1.8.1 Komponenty ANSI-RBAC – základní jádro RBAC

Obdobně jako model RBAC$_0$ RBAC96 poskytuje tato komponenta základní stavební bloky RBAC systému. V souboru základních prvků se nachází U (USERS – uživatelé), S (SESSIONS-sezení), R (ROLES – role) a PRMS[21] (PERMISSION – oprávnění) skládající se z OBS (objects – objekty) a OPS (operations – operace), vztahy PA (Permission Assignment – přiřazená oprávnění) a UA (User Assignment – přiřazení uživatele).



Obrázek 9 ANSI-RBAC – Základní jádro RBAC (INCITS 2004)

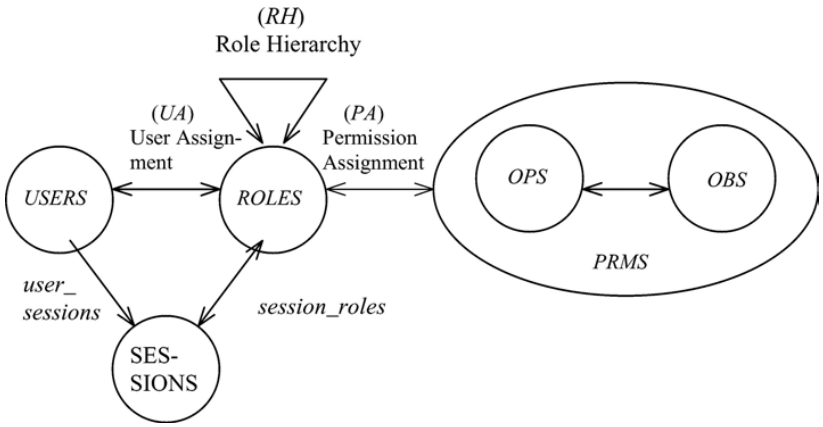
Obrázek 9 zobrazuje oboustranné šipky mezi USERS a ROLES, což značí vztah m:n, tedy že uživatel může mít více rolí a každá role může mít více uživatelů. Přiřazení oprávnění (PA) poukazuje na práva k souborům a adresářům např. právo na vytváření, čtení, zápis,

[21] Někdy je označováno jen písmenem „P“.

výmaz, modifikaci, apod. a jejich různé možné kombinace. Každé sezení SESSION mapuje jednoho uživatele a jeho role, které během sezení uživatel aktivoval. Každá SESSION je tak spojena s jedním uživatelem a každý uživatel je spojen s jednou nebo více SESSION. Funkce session_roles poskytuje role aktivované SESSION a funkce SESSION_USERS poskytuje uživatele, který je asociován se SESSION. Poskytovaná oprávnění uživatele jsou oprávnění přiřazená rolím, které má uživatel aktuálně aktivované skrz všechny jeho SESSION(s). (INCITS 2004)

1.8.2 Komponenty ANSI-RBAC – hierarchická RBAC

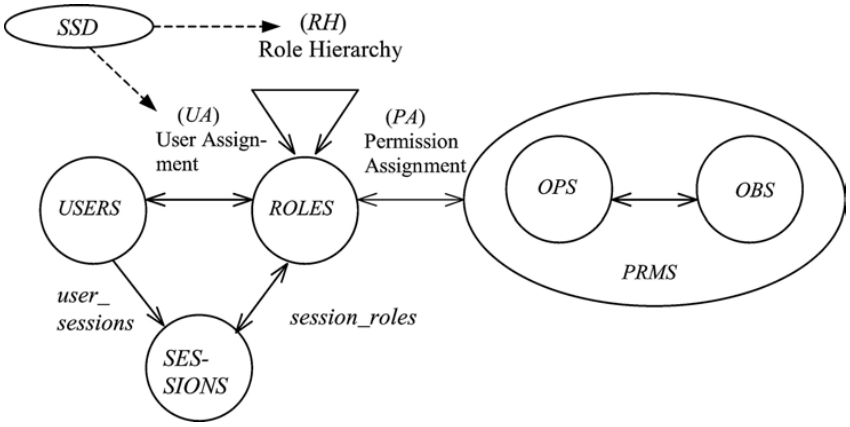
Hierarchie rolí je běžně používaná jako klíčový aspekt RBAC modelů a jsou často součástí konečných řešení s RBAC. Hierarchie jsou přirozeným prostředkem ke strukturování rolí tak odrážející obraz hierarchie organizační struktury. Mezi jednotlivými úrovněmi rolí vzniká tzv. vztah dědičnosti. Příkladem dědičnosti může být situace, kdy role r_1 dědí roli r_2, jestliže všechna oprávnění role r_2 jsou take oprávněními role r_1. (INCITS 2004)



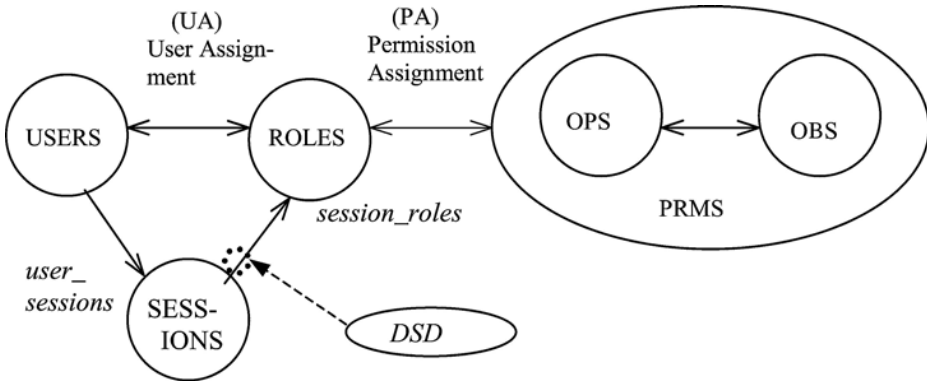
Obrázek 10 ANSI-RBAC – hierarchická RBAC (INCITS 2004)

1.8.3 Komponenty ANSI-RBAC – omezující RBAC

Omezující RBAC přidává oddělení úloh do modelu RBAC stejně jako model $RBAC_2$, a to i statické a dynamické (popsáno výše).



Obrázek 11 ANSI-RBAC – Statické oddělení úloh s hierarchickou RBAC (INCITS 2004)



Obrázek 12 ANSI-RBAC – Dynamické oddělení úloh RBAC (INCITS 2004)

1.9 Formální specifikace

Vzhledem k rozšířenějšímu uchopení a novějšímu přístupu k RBAC se práce bude zabývat výhradně syntaxí ANSI-RBAC, který používá Z notaci, která je standardizovanou notací ANSI a je definována v ISO/IEC 13568:2002 (Informační technologie – Z formální specifikace notace).

1.9.1 Komponenta ANSI-RBAC – základní jádro RBAC

$USERS, ROLES, OPS\ a\ OBS$ – uživatelé, role, příslušné operace a objekty.

$UA \subseteq USERS\ X\ ROLES$ – vztah m:n, zmapování přiřazení vztahu uživatel-role.

$assigned\_users: (r: ROLES) \rightarrow 2^{USERS}$ – zmapování role r na sadu uživatelů, jinak formálně $assigned\_users(r) = \{u \in USERS\ |(u, r)\ \in UA\}$

$PRMS = 2^{(OPS\ x\ OBS)}$ – sada oprávnění

$PA \subseteq PERMS\ X\ ROLES$ – vztah m:n, zmapování přiřazení vztahu oprávnění – role.

$assigned\_permissions(r: ROLES) \rightarrow 2^{PRMS}$ – zmapování role r na sadu oprávnění, jinak formálně $assigned\_permissions(r) = \{p \in PRMS \mid (p, r) \in PA\}$

$Op(p: PRMS) \rightarrow \{op \subseteq OPS\}$ – oprávnění do zmapování operací, tím lze získat sadu operací spojených s oprávněním p

$Ob(p: PRMS) \rightarrow \{ob \subseteq OBS\}$ – oprávnění do zmapování objektů, tím lze získat sadu objektů spojených s oprávněním p

$SESSIONS$ – sada sezení

$session\_users: (s: SESSIONS) \rightarrow USERS$ – zmapování sezení na odpovídající uživatele

$session\_roles: (s: SESSIONS) \rightarrow 2^{ROLES}$ – zmapování sezení na sadu rolí, jinak formálně $session\_roles(s_i) \subseteq \{r \in ROLES \mid (session\_users\ (s_i), r) \in UA\}$

$avail\_session\_perms(s: SESSIONS) \rightarrow 2^{PRMS}$ – oprávnění dostupná pro uživatele v sezení = $\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$

1.9.2 Komponenty ANSI-RBAC – hierarchická RBAC

Obecná hierarchie rolí:

$RH \subseteq ROLES\ X\ ROLES$ je částečně seřazen na $ROLES$ označovaný jako dědičnost vztahu, psaný jako $\geqslant$, kde $r_1 \geqslant r_2$ jen jestliže všechna oprávnění r_2 jsou také oprávněními r_1 a všichni uživatelé r_1 jsou také uživateli r_2, např.: $r_1 \geqslant r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions\ (r_1)$

$authorized\_users\ (r: ROLES) \rightarrow 2^{USERS}$ zmapování role r na sadu uživatelů představujících hierarchii rolí. Formálně $authorized\_users(r) = \{u \in USERS \mid r' \geqslant r, (u, r') \in UA\}$

$authorized\_permissions\ (r: ROLES) \rightarrow 2^{PRMS}$ zmapování role r na sadu oprávnění představujících hierarchii rolí. Formálně $authorized\_permissions(r) = \{p \in PRMS \mid r' \geqslant r, (p, r') \in PA\}$

Omezená hierarchie rolí:

Jedná se o obecnou hierarchii s následujícím omezením:

$$\forall\ r, r_1, r_2 \in ROLES, r \geqslant r_1 \wedge r \geqslant r_2 \Rightarrow r_1 = r_2$$

1.9.3 Komponenty ANSI-RBAC – omezující RBAC

Statické oddělení úloh (SSD)

$SSD \subseteq (2^{ROLES} X\ N)$ je kolekce párů (rs, n) ve statickém oddělení úloh, kde každá rs je sada rolí, (t) podmnožina rolí v rs a n je přirozené číslo ≥2, s vlastností, že žádný uživatel není přiřazen k n nebo vice rolích ze sady rs v které $(rs, n) \in SSD$. Formálně, $\forall (rs, n) \in SSD, \forall t\ \subseteq rs: |t|\ \geq n\ \Longrightarrow\ \bigcap_{r \in t} assigned\_users(r) = \emptyset$.

Statické oddělení úloh při výskytu v hierarchii

$\forall (rs, n) \in SSD, \forall t\ \subseteq rs: |t|\ \geq n\ \Longrightarrow\ \bigcap_{r \in t} authorized\_users(r) = \emptyset$.

Dynamické oddělení úloh (DSD)

$DSD \subseteq (2^{ROLES} X\ N)$ je kolekce párů (rs, n) v dynamickém oddělení úloh, kde každá rs je sada role a n je přirozené číslo ≥2, s vlastností, že žádný subjekt není aktivován v n nebo vice rolích ze sady rs ve které $dsd \in SSD$. Formálně:

$\forall rs \in 2^{ROLES}, n\ \in N, (rs, n) \in DSD\ \Longrightarrow\ n\ \geq 2\ .|rs| \geq n$

a

$\forall s \in \text{SESSIONS}, \forall rs \in 2^{ROLES}, \forall role_{subset} \in 2^{ROLES}, \forall n \in N, (rs, n) \in DSD,$

$role_{subset} \subseteq rs, role\_subset \subseteq session\_roles(s) \Longrightarrow |role?subset| < n$

1.10 RBAC a ABAC

Výzkumníci a vývojáři systému zjednodušili administrativní proces pomocí aplikování skupin uživatelů mající stejná oprávnění, uživatelské skupiny byly předzvěstí řízení přístupů na základě rolí. ABAC[22] - řízení přístupů na základě atributů neboli založený na vlastnostech, se zrodil opět v NIST (Hu et al., 2015). V současné době je formalizován (Hu a Kent, 2012, Hu et al., 2014) a očekává se jeho kompletní dokončení do roku 2019. ABAC používá označení objektů a uživatelských atributů místo oprávnění (Rajpoot et al., 2015). ABAC a RBAC se navzájem nevylučují naopak ABAC je v podstatě další rozšíření RBAC, kdy v rámci ABACu může role vystupovat jako další atribut, protože ABAC nepoužívá přístup přes role s oprávněními. (Coyne a Weil, 2013)

[22] Attribute-based access control

Dále se dle (Kuhn, 2011) zkoumá používání současných metod pro analýzu chybných hierarchií tak, aby bylo možné definovat zranitelnost v řízení přístupů, a zavádí se (Kern et al., 2004) taxonomie a podmínky detekce chybných hierarchií.

V dalším článku od (Kuhn et al., 2010), který se zabývá distribuovanými aplikacemi včetně atributů různých typů, dochází k popisu a posílení modelu RBAC díky rozšíření o atributy a poskytuje mechanismus zahrnující atributy v rozhodování při řízení přístupů.

2 Správa identit a algoritmy pro analýzu přístupových oprávnění

V rámci projektu *„Napojení systému pro správu identit na algoritmy pro analýzu přístupových oprávnění a modelování uživatelských rolí“*, který byl sub projektem *EC 5.1SPK01/013 - HRADECKÝ IT KLASTR (2009-2012, MPO/EC)* proběhl výzkum a vývoj provázání existujících systémů pro správu identit na externí komponentu, která bude zajišťovat návrh a životní cyklus systému rolí. Stěžejním výstupem byla analýza situace s oprávněními v již aktuálně provázaných systémech a aplikacích. Následně byla provedena formalizace přístupů do stávajících systémů a aplikací a tím zvýšení jejich vnitřní bezpečnosti při provozu. Dále proběhl vývoj vhodných algoritmů, které jsou použitelné k modelování katalogu uživatelských rolí, přičemž byly hledány vhodné omezující podmínky jako vstupní parametry jednotlivých algoritmů. Vybrané algoritmy byly implementovány nad existující databáze uživatelských kont, proto bylo nutné vyvinout konektory na katalogy rolí existujících systémů pro správu identit. V testovací fázi proběhlo praktické ověření využitelnosti zvolených algoritmů pro základní typy úloh.

2.1 Základní pravidla

Hlavním cílem modelování uživatelských rolí je dosažení nejlepšího a konečného počtu rolí. Stěžejním záměrem při modelování uživatelských rolí je dosažení takového stavu, kdy všichni uživatelé mají pouze taková oprávnění, která potřebují a zároveň nevlastní taková práva, která nepotřebují, resp. nemohou vlastnit. Z této jednoduché logické sekvence plyne základní pravidlo při modelování uživatelských rolí, a to: „Neměla by existovat práva, která nejdou popsat pomocí použití definovaných rolí.“

V rámci modelování rolí a pravidel jsou jednotlivá oprávnění (atributy) zahrnuty do organizačních rolí. Uživatel pak dostává příslušná oprávnění prostřednictvím svého přidělení k jednotlivým organizačním rolím. Pravidla pak definují zákonitosti a dále předělování jednotlivých organizačních rolí.

2.1.1 Jedna role pokrývá účet

Účet je pokryt jednou rolí, pokud pro všechny atributy, jež se mají pokrýt, platí, že role a účet mají shodné hodnoty u atributů stejných názvů.

(2-1)

$$\{Role_{nazev}: (atribut = (hodnota_{1,}\ hodnota_{2,}\ hodnota_{3,}\ \ldots . hodnota_{n}))\}$$

Příklad: V systému jsou takové účty, které mají jeden celočíselný atribut *celociselnyatribut* (datový typ integer). Doména tohoto atributu v rámci všech účtů v systému je 1-5. Zápis, kdy je požadována celá množina účtů ke kompletnímu pokrytí pomocí role:

(2-2)

$$\{Role_1: (celociselnyatribut = (1,2,3,4,5))\}$$

2.1.2 Sčítání rolí

Účet lze samozřejmě pokrýt i více než jednou rolí a pokud se pokrývá účet více rolemi, probíhá tzv. sčítání rolí, které pokrývají daný účet. Jedná se o významnou vlastnost modelu RBAC a slouží k redukci počtu rolí. Při sčítání rolí se každý atribut vyhodnocuje zvlášť v závislosti na jeho definici pro sčítání rolí.

(2-3)

$$\{Role_{nazev}: (atribut_{typ_{atributu}} = (hodnota_{1,}\ hodnota_{2,}\ hodnota_{3,}\ \ldots . hodnota_{n}))\}$$

2.1.3 Typy definic atributů dle chování při sčítání rolí

Typů definic atributů dle chování při sčítání rolí lze nalézt širokou škálu, vzhledem k využití používaných definic, se tato práce zabývá těmi nejčastějšími a nejpodstatnějšími:

- *Highest-value*
- *Multi-value*
- *Priority*

Samozřejmě využitím těchto nejpodstatnějších definic atributů se nepopírají logické operace AND, OR a XOR, dále lze analogicky definovat Minimum-value, ale i Průměr apod.

2.1.3.1 Highest-value

V součtu rolí dochází vždy k výběru právě takového atributu nesoucího nejvyšší hodnotu.

(2-4)

$$\{Role_{nazev}: (atribut_{HigestValue} = (hodnota))\}$$

Příklad: V systému jsou takové účty, které mají dva celočíselné atributy: $celociselnyatribut_1$ (4-6) a $celociselnyatribut_2$ (1-3). Oba tyto atributy jsou pro sčítání

rolí definovány jako *HighestValue* atributy, to znamená, že v součtu rolí dojde vždy k výběru atributu s nejvyšší hodnotou. Pokud se sečtou všechny 3 role, pokryjí účet atributu s hodnotami $celociselnyatribut_1$ (6) a $celociselnyatribut_2$. (3). Pro úplnost byly zmíněny ostatní možné kombinace.

(2-5)

$$\left\{Role_1 : \left[celociselnyatribut_{1_{HighestValue}} = (4)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (3)\right]\right\}$$

$$\left\{Role_2 : \left[celociselnyatribut_{1_{HighestValue}} = (5)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (2)\right]\right\}$$

$$\left\{Role_3 : \left[celociselnyatribut_{1_{HighestValue}} = (6)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (1)\right]\right\}$$

$$\{Role_1\} + \{Role_2\} + \{Role_3\}$$

$$= \left\{\left\{Role_{s(1+2+3)}\right\} : \left[celociselnyatribut_{1_{HighestValue}} = (6)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (3)\right]\right\}$$

$$\{Role_1\} + \{Role_2\}$$

$$= \left\{\left\{Role_{s(1+2)}\right\} : \left[celociselnyatribut_{1_{HighestValue}} = (5)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (3)\right]\right\}$$

$$\{Role_2\} + \{Role_3\}$$

$$= \left\{\left\{Role_{s(2+3)}\right\} : \left[celociselnyatribut_{1_{HighestValue}} = (6)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (2)\right]\right\}$$

$$\{Role_1\} + \{Role_3\}$$

$$= \left\{\left\{Role_{s(1+3)}\right\} : \left[celociselnyatribut_{1_{HighestValue}} = (6)\right]; \left[celociselnyatribut_{2_{HighestValue}} = (3)\right]\right\}$$

2.1.3.2 Multi-value

Jedná se o atributy obsahující více hodnot. U multi-value atributů nedochází k výběru jedné hodnoty, ale součet rolí je pokryt sjednocením všech hodnot atributu v rolích, které jsou sčítány. Důležité je zmínit, že v atributech nevznikají duplicity a nezáleží na pořadí hodnot v atributu.

(2-6)

$$\{Role_{nazev} : (atribut_{MultiValue} = (hodnota), .. n); (atribut_{MultiValue} = (hodnota), .. n)\}$$

Příklad 1: V systému jsou účty, které mají dva řetězcové *MultiValue* atributy – *atributA (X, R) a atributB (Y, Z).*

(2-7)

$$\{Role_1 : (atributA_{MultiValue} = (X)); (atributB_{MultiValue} = (Y))\}$$

$$\{Role_2 : (atributA_{MultiValue} = (R)); (atributB_{MultiValue} = (Z))\}$$

$$\{Role_1\} + \{Role_2\}$$

$$= \left\{\{Role_{s(1+2)}\} : (atributA_{MultiValue} = (X, R)); (atributB_{MultiValue} = (Y, Z))\right\}$$

Příklad 2: V systému jsou účty, které mají jeden řetězcový *MultiValue* atribut – *atributA* (R, X, Y, Z). Z následujících rolí bude zjišťováno, co pokrývají:

(2-8)

$$\{Role_1 : (atributA_{MultiValue} = (R, X, Y, Z))\}$$

$$\{Role_2 : (atributA_{MultiValue} = (X, Y, Z, E))\}$$

$$\{Role_1\} + \{Role_2\}$$

$$= \left\{\{Role_{s(1+2)}\} : (atributA_{MultiValue} = (R, X, Y, Z, E))\right\}$$

Žádný jiný účet součet rolí $Role_1 + Role_2$ nepokrývá, což vyplývá z unikátnosti hodnot.

2.1.3.3 Priority

Definice atributů typu priority říká, že každá role má prioritu. Při součtu rolí dochází k výběru u atributů typu priority tak, že výsledkem jsou takové hodnoty, které mají nejvyšší prioritu. V případě více rolí se stejnou prioritou atributu typu priority mající různé hodnoty, pak nelze součet těchto rolí s danými atributy pokrýt (nelze pokrýt žádný účet v systému). Jestliže jsou hodnoty u atributu typu priority stejné, pokrývají danou stejnou hodnotu.

(2-9)

$$\{Role_{nazev}(Priority(1..n)) : (atribut_{Priority} = (hodnota))\}$$

Příklad 1: V systému jsou účty, které mají tři celočíselné atributy $celociselnyatribut_1$ (5-6) a $celociselnyatribut_2$.(3-4) a $celociselnyatribut_3$.(1-2). Atribut $celociselnyatribut_1$ je typu HighestValue, ostatní atributy jsou typu Priority.

(2-10)

$$\left\{Role_1[Priority(5)]: \left[celociselnyatribut_{1_{HighestValue}} = (6)\right]; \left[celociselnyatribut_{2_{Priority}} = (4)\right]; \left[celociselnyatribut_{3_{Priority}} = (1)\right]\right\}$$

$$\left\{Role_2[Priority(8)]: \left[celociselnyatribut_{1_{HighestValue}} = (5)\right]; \left[celociselnyatribut_{2_{Priority}} = (3)\right]; \left[celociselnyatribut_{3_{Priority}} = (2)\right]\right\}$$

$$\{Role_1\} + \{Role_2\}$$

$$= \left\{\{Role_{s(1+2)}\}: \left[celociselnyatribut_{1_{HighestValue}} = (6)\right]; \left[celociselnyatribut_{2_{Priority}} = (3)\right]; \left[celociselnyatribut_{3_{Priority}} = (2)\right]\right\}$$

Příklad 2: V systému jsou účty, které mají jeden číselný atribut $celociselnyatribut_1$ (5-6), který je typu Priority.

(2-11)

$$\left\{Role_1[Priority(5)]: \left[celociselnyatribut_{1_{Priority}} = (6)\right]\right\}$$

$$\left\{Role_1[Priority(5)]: \left[celociselnyatribut_{1_{Priority}} = (5)\right]\right\}$$

$$\{Role_1\} + \{Role_2\}$$

$$= \left\{\{Role_{s(1+2)}\}: \left[celociselnyatribut_{1_{Priority}} = (\emptyset)\right]\right\}$$

$Role_1 + Role_2$ nepokrývají žádný účet v systému.

Příklad 3: V systému jsou účty, které mají jeden číselný atribut $celociselnyatribut_1$ (5), který je typu Priority.

(2-12)

$$\left\{Role_1[Priority(5)]: \left[celociselnyatribut_{1_{Priority}} = (5)\right]\right\}$$

$$\left\{Role_1[Priority(5)]: \left[celociselnyatribut_{1_{Priority}} = (5)\right]\right\}$$

$$\{Role_1\} + \{Role_2\}$$

$$= \left\{\{Role_{s(1+2)}\}: \left[celociselnyatribut_{1_{Priority}} = (5)\right]\right\}$$

$Role_1 + Role_2$ pokrývají účet s atributem $celociselnyatribut_{1_{Priority}} = (5)$, což vyplývá ze shodné priority.

2.2 Formální specifikace modelování rolí

Pro vývoj aplikace bylo dále nutné formalizovat základní specifikaci modelování pro využití algoritmů Hill Climb, genetického algoritmu a SAT algoritmu (Simkova a Poulova, 2012).

2.2.1 Základní datové typy

$ATTRIBS\_NAMES$ je množina všech textových řetězců, které mohou tvořit názvy jednotlivých atributů a $ATTRIBS\_TYPE$ tvoří množinu všech typů hodnot atributů při sčítání rolí (THPSR). Jedná se o typy, podle nichž se budou řídit algoritmy při sčítání rolí, jsou definované 4 datové typy *(2-13)*.

(2-13)

$$ATTRIBS\_TYPE = \{HighestValue, MultiValue, Priority, Undefined\}$$

$ATTRIBS\_VALUES$ představují množinu všech přípustných hodnot typů THPSR a $ATTRIBS\_VALUES\_SET$ jsou všechny množiny přípustných hodnot jednotlivých atributů, každá množina hodnot má stejný THPSR *(2-14)*.

(2-14)

$$ATTRIBS\_VALUES\_SET \subseteq POW(ATTRIBS\_VALUES)$$

2.2.2 Atribut

Představuje složený objekt a je nutné se vždy dotázat konkrétního atributu na uvedené druhy informací:

- Název, který je vyjádřen textovým řetězcem.
- Typy hodnot atributů při sčítání rolí (THPSR).
- Množinu hodnot, které odpovídají THPSR.

$ATTRIBS$ tedy přestavuje celou množinu všech přípustných atributů, přičemž, jak již bylo popsáno v kapitole výše, rozlišují se $HV\_ATTRIBS$ pro množinu všech atributů, jejichž THPSR je $HigestValue$, $MV\_ATTRIBS$ pro množinu všech atributů, jejichž THPSR je MultiValue a $P\_ATTRIBS$ pro množinu všech atributů, jejichž THPSR je Priority.

(2-15)

$$getAttribName: ATTRIBS \rightarrow ATTRIBS\_NAMES$$

$$getAttribValues: ATTRIBS \rightarrow ATTRIBS\_VALUES\_SET$$

$$getAttribType: ATTRIBS \rightarrow ATTRIBS\_TYPE$$

Popis *(2-15)* je následující $getAttribName$, $getAttribValues$ a $getAttribType$ představují funkce zobrazení odpovídající svému jménu, přičemž $getAttribType$ lze měnit v závislosti na požadavcích ke konkrétním atributům, které se mají pokrýt. V případě výběru konkrétního atributu do pokrývání nesmí být hodnota $getAttribType$ pro $\{Undefined\}$, jinak by tento atribut do procesu pokrývání zahrnut nebyl.

2.2.2.1 Axiomy

(2.2.1)

$$HV\_ATTRIBS = \left\{ \begin{array}{l} attrib \in ATTRIBS : getAttribType(attrib) = HighestValue, \\ \left| getAttribValues(attrib) \right| \le 1 \end{array} \right\}$$

$$\forall attrib \in HV\_ATTRIBS ; getAttribValues(attrib) \neq \{\ \} \Rightarrow getAttribValues(attrib) \in R$$

$$MV\_ATTRIBS = \{attrib \in ATTRIBS : getAttribType(attrib) = MultiValue\}$$

$$P\_ATTRIBS = \left\{ \begin{array}{l} attrib \in ATTRIBS : getAttribType(attrib) = Priority, \\ \left| getAttribValues(attrib) \right| \le 1 \end{array} \right\}$$

$$\forall attrib \in DOM(getAttribType); attrib \notin \{HV\_ATTRIBS \cup MV\_ATTRIBS \cup P\_ATTRIBS\} \Rightarrow$$

$$getAttribType(attrib) = \{Undefined\}$$

$$\forall (attrib1, attrib2) \in ATTRIBS ; getAttribName(attrib1) = getAttribName(attrib2) \Rightarrow$$

$$getAttribType(attrib1) = getAttribType(attrib2)$$

2.2.3 Uživatelský účet

Uživatelský účet $ACCOUNTS$ je zde chápán jako seskupení atributů utvořeného z dat, načtených přímo z LDAP[23] nebo LDIF[24]. Na každý jednotlivý atribut určitého účtu se lze odkázat prostřednictvím textového řetězce, který představuje název tohoto atributu, obecně viz *(2-16)*. Pokud pro specifikovaný textový řetězec neexistuje v účtu žádný

[23] Lightweight Directory Access Protocol – přístupový protokol mezi klientem a adresářovým serverem (X. 500) na bázi TCP/IP.
[24] LDIF – standardizovaný formát, který reprezentuje data adresářového serveru jako množinu záznamů v textové formě.

atribut stejného názvu, potom je výsledkem dotazu prázdná množina. $ACCOUNTS$ tedy obsahuje množinu všech přístupných účtů.

(2-16)

$$getAttrib: ACCOUNTS\ x\ ATTRIBS\_NAMES \rightarrow ATTRIBS$$

Samotný algoritmus vyžaduje pro tvorbu tzv. agregovaných účtů (viz dále) prostředky pro porovnání dvojice účtů, které zjistí, zda jsou tyto účty v jistém smyslu ekvivalentní *(2-17).* Dva účty jsou ekvivalentní právě tehdy, když jsou ekvivalentní všechny dvojice příslušných atributů k pokrytí.

(2-17)

$$isAccountsEqual: ACCOUNTS\ x\ ACCOUNTS \rightarrow \{true, false\}$$

Dvojice atributů je ekvivalentní právě tehdy *(2-18)*, když mají oba atributy shodný název, THPSR a množiny hodnot.

(2-18)

$$isAttribsEqual: ATTRIBS\ x\ ATTRIBS \rightarrow \{true, false\}$$

Atributy, jejichž THPSR je pro $\{Undefined\}$ se ignorují, což je smysluplné, neboť vzájemné porovnávání atributů má dobrý význam pouze tehdy, když uživatel již definoval atributy k pokrytí. Množina názvů všech atributů k pokrytí je označena jako $ATTRIBS\_NAMES\_TO\_COVER$.

2.2.3.1 Axiomy

(2.2.2)

$$\begin{aligned}
&\forall(attrib1, attrib2) \in DOM(isAttribsEqual); isAttribsEqual(attrib1, attrib2) = true \Leftrightarrow \\
&(getAttribName(attrib1) = getAttribName(attrib2)) \\
&\wedge (getAttribType(attrib1) \neq \{Undefined\}) \\
&\wedge (getAttribType(attrib1) = getAttribType(attrib2)) \\
&\wedge (getAttribValues(attrib1) = getAttribValues(attrib2))
\end{aligned}$$

$$\begin{aligned}
&\forall(account1, account2) \in DOM(isAccountsEqual); \\
&isAccountsEqual(account1, account2) = true \Leftrightarrow \\
&\left(\begin{array}{l} \forall attribName \in ATTRIBS_NAME_TO_COVER; \\ isAttribsEqual(getAttrib(account1, attribName), getAttrib(account2, attribName)) = true \end{array}\right)
\end{aligned}$$

2.2.4 Agregovaný účet

Zobrazení $isAccountsEqual$ tvoří na konkrétní množině účtů jednotlivé třídy ekvivalence. Pro účely algoritmu hledání pokrytí lze každou takovou třídu ekvivalence nahradit novým, speciálně vytvořeným účtem, jehož množina názvů všech jeho atributů je totožná s množinou názvů všech atributů k pokrytí *(2-19).* Předpokladem pro tvorbu agregovaných účtů je tedy existence množiny vstupních účtů a existence zobrazení $isAccountsEqual$. $INPUT\_ACCOUNTS$ tak představuje množinu platných vstupních účtů a $AGREGATED\_ACCOUNTS$ množinu agregovaných účtů příslušnou množině.

(2-19)

$$getAgregatedAccount: INPUT\_ACCOUNTS \rightarrow AGREGATED\_ACCOUNTS$$

2.2.4.1 Axiomy

(2.2.3)

$$INPUT\_ACCOUNTS \subset ACCOUNTS$$

$$AGREGATED\_ACCOUNTS \subset ACCOUNTS$$

$$\forall account \in AGREGATED\_ACCOUNTS, \forall attribName \notin ATTRIBS\_NAME\_TO\_COVER; getAttrib(account, attribName) = \{\ \}$$

$$\forall (account1, account2) \in INPUT\_ACCOUNTS;$$

$$isAccountsEqual(account1, account2) = true \Rightarrow$$

$$getAgregatedAccount(account1) = getAgregatedAccount(account2)$$

$$\forall account \in INPUT\_ACCOUNTS;$$

$$isAccountsEqual(account, getAgregatedAccount(account)) = true$$

2.2.5 Role

Stejně jako uživatelský účet, je též role seskupení jistých atributů $ROLES\_SETS$ je tvořen takovou množinou, která obsahuje všechny množiny rolí. Podstatný rozdíl je však v tom, že hodnoty těchto atributů nejsou známy (kromě rolí předdefinovaných[25]). Úkolem algoritmů pokrytí je právě přesně specifikovat atributy rolí tak, aby byly splněny jisté požadavky na pokrytí vstupní množiny účtů.

[25] Předdefinovaná role je taková role, která bude součástí následující modelované sady rolí.

Další rozdíl oproti definici uživatelského a agregovaného účtu je v tom, že každá role má přiřazenou jistou prioritu, kterou musí respektovat algoritmy pokrývání, viz dále. Role, které byly vytvořeny některým z algoritmů pokrývání, mají definitoricky prioritu nula. Předdefinované role, které byly známy již před vlastním výpočtem algoritmů pokrývání, mají prioritu stanovenou na základě uživatelova rozhodnutí *(2-20)*.

(2-20)

$$getPriority = \{role: \exists roles \in ROLES\_SETS; role \in roles \rightarrow Z$$

2.2.5.1 Axiomy

(2.2.4)

$$\forall roles \in ROLES\_SETS; \forall role \in roles; role \in ACCOUNTS$$

$$\forall (role, attribName) \in \left\{ \begin{array}{c} (role\_y, attribName\_z): \\ roles\_x \in ROLES\_SETS, \\ role\_y \in roles\_x, \\ attribName\_z \in ATTRIBS\_NAME\_TO\_COVER \end{array} \right\}$$

$$getAttrib(role, attribName) \neq \{\ \} \wedge getAttribType(getAttrib(role, attribName)) \neq \{Undefined\}$$

$$\forall (roles, role, attribName) \in \left\{ \begin{array}{c} (role\_y, attribName\_z): \\ roles\_x \in ROLES\_SETS, \\ role\_Y \in roles\_x, \\ attribName\_z \in ATTRIBS\_NAME\_TO\_COVER \end{array} \right\}$$

$$getAttrib(role, attribName) = \{\ \}$$

2.2.6 Pokrývání účtů rolemi

Účet je pokryt danou rolí, pokud pro všechny atributy, které mají být pokryty, platí, že role a účet mají stejné hodnoty u atributů stejných názvů. Častěji se však vyskytuje případ, kdy se pro pokrytí jednoho účtu používá rolí více. Účet je danou množinou rolí pokryt právě tehdy, když je touto množinou rolí pokryt každý jeho atribut poté, co se všechny role této množiny sečtou. Každý atribut každé role z této množiny rolí musí být s odpovídajícím atributem stejného názvu účtu k pokrytí v jisté relaci *(2-21)*, *(2-22)*.

(2-21)

$$getHVRelation: HV\_ATTRIBS\ x\ HV\_ATTRIBS \rightarrow \{Lower, Equal, Higher\}$$

(2-22)

$$isRoleAdeptToCoverAttrib:$$

$$\{role: \exists roles \in ROLES\_SETS; role \in roles\}\ x\ ATTRIBS \rightarrow \{true, false\}$$

Každou roli původní množiny rolí, která splňuje tyto relace, lze zařadit do tzv. množiny adeptů na pokrytí daného účtů *(2-22)*. Pokud daná role nepatří do množiny adeptů daného účtu, potom nelze tuto roli k pokrytí daného účtu vůbec použít. Protože jsou jednotlivé účty k pokrytí obecně různé *(2-23)*, budou různé i množiny jejich adeptů k pokrytí

(2-23)

$$isRoleAdeptToCoverAccount:$$

$$\{role: \exists roles \in ROLES\_SETS; role \in roles\}\ x\ AGREGATED\_ACCOUNTS \rightarrow \{true, false\}$$

O existenci pokrytí atributu daného účtu množinou svých adeptů lze rozhodnout samostatně *(2-24)*.

(2-24)

$$isAttribCoveredByRoles:$$

$$\left\{ \begin{matrix} (account, attrib): account \in AGREGATED\_ACCOUNTS, \\ attrib \neq \{\}, attrib = getAttrib(account, getAttribName(attrib)) \end{matrix} \right\}\ x\ ROLES\_SETS \rightarrow \{true, false\}$$

Vlastní algoritmus rozhodnutí závisí primárně na THPSR daného atributu účtu, atributů stejného názvu množiny adeptů účtu a prioritě jednotlivých rolí z této množiny adeptů *(2-25)*.

(2-25)

$$isPriorAttribsCoveredByRoles: AGREGATED\_ACCOUNTS\ x\ ROLES\_SETS \rightarrow \{true, false\}$$

$$isAccountCoveredByRoles: AGREGATED\_ACCOUNTS\ x\ ROLES\_SETS \rightarrow \{true, false\}$$

2.2.6.1 Axiomy

Pokrytí rolemi pro atributy typu $HigestValue$:

(2.2 5)

$$\begin{aligned}
&\forall (account, attrib, roles) \in DOM(isAttribCoveredByRoles), attrib \in HV_ATTRIBS; \\
&\left(isAttribCoveredByRoles(account, attrib, roles) = true\right) \Leftrightarrow \\
&\left(\begin{matrix} \left(\forall role \in roles; isRoleAdeptToCoverAccount(role, account) = true\right) \\ \wedge \left(\begin{matrix} \exists roleEq \in roles, \exists roleEqAttrib \in HV_ATTRIBS; \\ roleEqAttrib = getAttrib(roleEq, getAttribName(attrib)) \wedge \\ getHVRelation(roleEqAttrib, attrib) = \{Equal\} \end{matrix} \right) \end{matrix} \right)
\end{aligned}$$

Pokrytí rolemi pro atributy typu $MultiValue$:

(2.2.6)

$$\begin{aligned}
&\forall(account, attrib, roles) \in DOM(isAttribCoveredByRoles), attrib \in MV_ATTRIBS;\\
&\left(isAttribCoveredByRoles(account, attrib, roles) = true\right) \Leftrightarrow\\
&\left(\left(\forall role \in roles; isRoleAdeptToCoverAccount(role, account) = true\right)\right.\\
&\left.\wedge\left(\bigcup_{roleValues}\left\{\begin{array}{l} roleValues : role \in roles, roleAttrib \in MV_ATTRIB,\\ roleAttrib = getAttrib(role, getAttribName(attrib)),\\ roleValues = getAttribValues(roleAttrib)\end{array}\right\}\right.\right.\\
&\left.\left.= getAttribValues(attrib)\right)\right)
\end{aligned}$$

Pokrytí rolemi pro všechny atributy typu $Priority$:

(2.2.7)

$$\begin{aligned}
&\forall(account, roles) \in DOM(isPriorAttribsCoveredByRoles);\\
&is\,\Pr ior AttribsCoveredByRoles(account, roles) = true \Leftrightarrow\\
&\left(\left(\forall role \in roles; isRoleAdeptToCoverAccount(role, account) = true\right) \wedge\right.\\
&\left(\left(\forall prior_attrib \in \left\{\begin{array}{l} pr_attr :\\ attribName \in ATTRIBS_TO_COVER,\\ pr_attr = getAttrib(account, attribName),\\ pr_attr \in P_ATTRIBS\end{array}\right\},\right.\right.\\
&\exists role \in roles;\\
&\left(\begin{array}{l} getAttribValues(getAttrib(role, getAttribName(prior_attrib))) =\\ = getAttribValues(prior_attrib)\end{array}\right)\\
&\left.\left.\wedge\neg\left(\begin{array}{l}\exists roleX \in roles;\\ getPriority(roleX) > getPriority(role) \wedge\\ isRoleAdeptToCoverAttrib(roleX, prior_attrib) = false\end{array}\right)\right)\right)
\end{aligned}$$

Pokrytí jednoho účtu:

(2.2.8)

$$\begin{aligned}
&\forall(account, roles) \in DOM(isAccountCoveredByRoles);\\
&isAccountCoveredByRoles(account, roles) \Leftrightarrow\\
&\left(\begin{array}{l}\forall attrib \in \left\{\begin{array}{l} attrib_nop :\\ attribName \in ATTRIBS_TO_COVER,\\ attrib_nop = getAttrib(account, attribName),\\ attrib_nop \notin P_ATTRIBS\end{array}\right\};\\ isAttribCoveredByRoles(attrib, roles) = true\end{array}\right)\\
&\wedge\\
&\left(isPriorAttribsCoveredByRoles(account, roles) = true\right)
\end{aligned}$$

Platí pro případy, kdy minimálně jeden z atributů typu $HigestValue$ má prázdnou množinu hodnot:

(2.2.9)

$$\forall(emptyAttrib1, emptyAttrib2) \in HV\_ATTRIBS; \begin{pmatrix} getAttribValues(emptyAttrib1) = \{\ \} \wedge \\ getAttribValues(emptyAttrib2) = \{\ \} \end{pmatrix} \Rightarrow$$
$$getHVRelation(emptyAttrib1, emptyAttrib2) = \{Equal\}$$

$$\forall(emptyAttrib, attrib) \in HV\_ATTRIBS; \begin{pmatrix} getAttribValues(attrib) \neq \{\ \} \wedge \\ getAttribValues(emptyAttrib) = \{\ \} \end{pmatrix} \Rightarrow$$
$$getHVRelation(emptyAttrib, attrib) = \{Lower\}$$

$$\forall(attrib, emptyAttrib) \in HV\_ATTRIBS; \begin{pmatrix} getAttribValues(attrib) \neq \{\ \} \wedge \\ getAttribValues(emptyAttribg) = \{\ \} \end{pmatrix} \Rightarrow$$
$$getHVRelation(attrib, emptyAttrib) = \{Higher\}$$

V případech, kdy oba atributy typu $HigestValue$ mají právě jednu hodnotu:

(2.2.10)

$$\forall(attrib1, attrib2) \in HV\_ATTRIBS; \begin{pmatrix} getAttribValues(attrib1) \neq \{\ \} \wedge \\ getAttribValues(attrib2) \neq \{\ \} \end{pmatrix} \Rightarrow$$
$$\left(getHVRelation(attrib1, attrib2) = \{Lower\} \Leftrightarrow getAttribValues(attrib1) < getAttribValues(attrib2)\right)$$

$$\forall(attrib1, attrib2) \in HV\_ATTRIBS; \begin{pmatrix} getAttribValues(attrib1) \neq \{\ \} \wedge \\ getAttribValues(attrib2) \neq \{\ \} \end{pmatrix} \Rightarrow$$
$$\left(getHVRelation(attrib1, attrib2) = \{Equal\} \Leftrightarrow getAttribValues(attrib1) = getAttribValues(attrib2)\right)$$

$$\forall(attrib1, attrib2) \in HV\_ATTRIBS; \begin{pmatrix} getAttribValues(attrib1) \neq \{\ \} \wedge \\ getAttribValues(attrib2) \neq \{\ \} \end{pmatrix} \Rightarrow$$
$$\left(getHVRelation(attrib1, attrib2) = \{Higher\} \Leftrightarrow getAttribValues(attrib1) > getAttribValues(attrib2)\right)$$

Rozhodnutí, zda je role adeptem na pokrytí atributu typu $HigestValue$:

(2.2.11)

$$\forall(role, attrib) \in DOM(isRoleAdeptToCoverAttrib);$$
$$\begin{pmatrix} attrib \in HV\_ATTRIBS \wedge \\ getAttribName(attrib) \in ATTRIBS\_NAMES\_TO\_COVER \end{pmatrix} \Rightarrow$$
$$\begin{pmatrix} isRoleAdeptToCoverAttrib(role, attrib) = true \Leftrightarrow \\ getHVRelation(getAttrib(role, getAttribName(attrib)), attrib) \in \{Lower, Equal\} \end{pmatrix}$$

Rozhodnutí, zda je role adeptem na pokrytí atributu typu $MultiValue$:

(2.2.12)

$$\forall(role, attrib) \in DOM(isRoleAdeptToCoverAttrib);$$
$$\left(\begin{array}{l} attrib \in MV\_ATTRIBS \wedge \\ getAttribName(attrib) \in ATTRIBS\_NAMES\_TO\_COVER \end{array}\right) \Rightarrow$$
$$\left(\begin{array}{l} isRoleAdeptToCoverAttrib(role, attrib) = true \Leftrightarrow \\ getAttribValues(getAttrib(role, getAttribName(attrib))) \subseteq getAttribValues(attrib) \end{array}\right)$$

Rozhodnutí, zda je role adeptem na pokrytí atributu typu $Priority$:

(2.2.13)

$$\forall(role, attrib) \in DOM(isRoleAdeptToCoverAttrib);$$
$$\left(\begin{array}{l} attrib \in P\_ATTRIBS \wedge \\ getAttribName(attrib) \in ATTRIBS\_NAMES\_TO\_COVER \end{array}\right) \Rightarrow$$
$$\left(\begin{array}{l} isRoleAdeptToCoverAttrib(role, attrib) = true \Leftrightarrow \\ \left(getAttribValues(getAttrib(role, getAttribName(attrib))) = getAttribValues(attrib)\right) \vee \\ \left(getAttribValues(getAttrib(role, getAttribName(attrib))) = \{\ \}\right) \end{array}\right)$$

Rozhodnutí, zda je role adeptem na pokrytí neprioritních atributů daného agregovaného účtu:

(2.2.14)

$$\forall(role, account) \in DOM(isRoleAdeptToCoverAccount);$$
$$isRoleAdeptToCoverAccount(role, account) = true \Leftrightarrow$$
$$\left(\begin{array}{l} \forall attrib \in \left\{\begin{array}{l} attrib\_nop: \\ attribName \in ATTRIBS\_NAMES\_TO\_COVER, \\ attrib\_nop = getAttrib(account, attribName), \\ attrib\_nop \notin P\_ATTRIBS \end{array}\right\}; \\ isRoleAdeptToCoverAttrib(role, attrib) = true \end{array}\right)$$

2.2.7 Fixovaný atribut

Uživatel má možnost tzv. fixovat atribut[26], kde $FIXED\_ATTRIB\_NAME$ je název fixovaného atributu a $FIX\_ATTR\_ROLES\_SETS$ představuje množinu rolí, které fixují definovaný atribut $FIXED\_ATTRIB\_NAME$.

K vysvětlení fixace atributu je třeba vysvětlit termín doména atributu. Doména atributu je sjednocení hodnot všech atributů stejného názvu množiny agregovaných účtů. Pokud

[26] Při fixaci atributu se vygenerují role, které budou pokrývat všechny hodnoty tohoto atributu.

má být atribut daného názvu fixován, potom musí výsledná množina rolí obsahovat v hodnotách svých atributů stejného názvu, jako je název fixovaného atributu, všechny hodnoty domény fixovaného atributu *(2-26)*. Navíc musí pro každou jednotlivou hodnotu z domény fixovaného atributu existovat role, která obsahuje právě pouze tuto hodnotu.

(2-26)

$$getAttribDomain: ATTRIBS\_NAMES\_TO\_COVER \rightarrow POW(ATTRIBS\_VALUES)$$

2.2.7.1 Axiomy

(2.2.15)

$$FIXED\_ATTRIB\_NAME \in ATTRIBS\_NAMES\_TO\_COVER$$

$$FIX\_ATTR\_ROLES\_SETS \subseteq ROLES\_SETS$$

$$\forall attribName \in DOM(getAttribDomain);$$
$$getAttribDomain(attribName) = \bigcup_{account \in AGREGATED\_ACCOUNT} getAttribValues(getAttrib(account, attribName))$$

$$\forall fixed\_roles \in FIX\_ATTR\_ROLES\_SETS;$$
$$getAttribDomain(FIXED\_ATTRIB\_NAME) \subseteq \bigcup_{role \in fixed\_roles} getAttribValues(getAttrib(role, FIXED\_ATTRIB\_NAME))$$

$$\forall fixed\_roles \in FIX\_ATTR\_ROLES\_SETS;$$
$$\begin{pmatrix} \forall attrib\_value \in getAttribDomain(FIXED\_ATTRIB\_NAME); \\ \exists role \in fixed\_roles; getAttribValues(getAttrib(role, FIXED\_ATTRIB\_NAME)) = attribValue \end{pmatrix}$$

2.2.8 Předdefinované role

Uživatel má možnost stanovit si tzv. předdefinované role $PREDEFINED\_ROLES$. Množina předdefinovaných rolí bude podmnožinou množiny vygenerovaných rolí.

2.2.8.1 Axiomy

(2.2.16)

$$PREDEFINED\_ROLES \in ROLES\_SETS$$

2.2.9 Maximalizační úloha

Úlohou maximalizačního algoritmu *(2-27)* je pokusit se vytvořit uživatelem stanovený počet rolí tak, aby tyto role pokryly ve zvolených atributech podmnožinu maximální kardinality z množiny všech vstupních účtů (Simkova a Tomaskova, 2012).

(2-27)

$$totalCoveredAccounts: ROLES\_SETS \rightarrow \mathbb{N} \cup \{0\}$$

Vstup:

1. Množina vstupních účtů - $INPUT\_ACCOUNTS$.
2. Množina všech názvů všech atributů k pokrytí - $ATTRIBS\_NAMES\_TO\_COVER$.
3. THPSR pro všechny atributy všech účtů z $INPUT\_ACCOUNTS$, jejichž název patří do množiny $ATTRIBS\_NAMES\_TO\_COVER$.
4. Počet rolí, které algoritmus vygeneruje, nechť je tento počet označen jako $rolesNum$.
5. Příležitostně: název fixovaného atributu - $FIXED\_ATTRIB\_NAME$.
6. Příležitostně: množina předdefinovaných rolí - $PREDEFINED\_ROLES$.

Optimální výstup:

1. Množina rolí, která pokrývá ve zvolených atributech podmnožinu maximální kardinality množiny $INPUT\_ACCOUNTS$, nechť je tato množina označena jako MAX_FINAL_ROLES.

2.2.9.1 Axiomy

(2.2.17)

$$\forall roles \in DOM(totalCoveredAccounts); totalCoveredAccounts(roles) = \left| \left\{ \begin{array}{l} accounts \in INPUT\_ACCOUNTS: \\ \exists cov\_roles \subseteq roles, \\ isAccountCoveredByRoles(getAgregatedAccount(account), cov\_roles) = true \end{array} \right\} \right|$$

$$MAX\_FINAL\_ROLES \in ROLES\_SETS$$

$$\left| MAX\_FINAL\_ROLES \right| = rolesNum$$

Pro fixovaný atribut:

(2.2.18)

$$MAX\_FINAL\_ROLES \in FIX\_ATTR\_ROLES\_SETS$$

Pro předdefinované role:

(2.2.19)

$$PREDEFINED\_ROLES \subseteq MAX\_FINAL\_ROLES$$

$$totalCoveredAccounts(MAX\_FINAL\_ROLES) = \max\left\{\begin{matrix} totalCoveredAccounts(roles): \\ roles \in DOM(totalCoveredAccounts), \\ |roles| = rolesNum \end{matrix}\right\}$$

2.2.10 Minimalizační úloha

Úlohou minimalizačního algoritmu je pokusit se vytvořit minimální počet rolí tak, aby tyto role pokryly ve zvolených atributech podmnožinu alespoň minimální, uživatelem definované kardinality množiny všech vstupních účtů *(2-27).*

Vstup:

1. Množina vstupních účtů - $INPUT\_ACCOUNTS$.
2. Množina všech názvů všech atributů k pokrytí - $ATTRIBS\_NAMES\_TO\_COVER$.
3. THPSR pro všechny atributy všech účtů z - $INPUT\_ACCOUNTS$, jejichž název patří do množiny $ATTRIBS\_NAMES\_TO\_COVER$.
4. Minimální počet účtů, které musí vygenerovaná množina rolí pokrýt; nechť je tento počet označen jako $covederMin$. V praxi není uveden přímo počet, ale jisté procento z velikosti množiny vstupních účtů.
5. Příležitostně: název fixovaného atributu - $FIXED\_ATTRIB\_NAME$.
6. Příležitostně: množina předdefinovaných rolí - $PREDEFINED\_ROLES$,

Optimální výstup:

1. Množina o takovém minimálním počtu rolí, která pokrývá ve zvolených atributech podmnožinu alespoň minimální, uživatelem definované kardinality množiny $INPUT\_ACCOUNTS$, nechť je tato množina označena jako MIN_FINAL_ROLES.

2.2.10.1 Axiomy

(2.2.20)

$$MIN\_FINAL\_ROLES \in ROLES\_SETS$$

Pro fixovaný atribut:

(2.2.21)

$$MIN\_FINAL\_ROLES \in FIX\_ATTR\_ROLES\_SETS$$

Pro předdefinované role:

(2.2 22)

$$PREDEFINED\ \_ROLES \subseteq MIN\ \_FINAL\ \_ROLES$$

$$\left|MIN\ \_FINAL\ \_ROLES\right| = \min\left\{\begin{array}{l} rolesSize: \\ roles \in DOM\,(totalCoveredAccounts), \\ rolesSize = \left|roles\right|, \\ totalCoveredAccounts(roles) \geq coveredMin \end{array}\right\}$$

2.3 Použité algoritmy

Aplikace načte[27] účty z LDIF nebo LDAP jednotného formátu, kdy uživatel si vybere atributy, které budou klíčové pro generování rolí, a nadefinuje pravidla pro sčítání rolí, kdy se rozhoduje mezi maximálním pokrytím účtů a minimálním tedy alespoň X% a dále lze volit tzv. fixaci rolí na zvolený atribut. V posledním případě pak počet rolí bude právě roven nebo vyšší než je doména tohoto atributu. Analýza rolí tedy bude probíhat s tím, že dochází k vyhodnocování kombinací zbývajících atributů. (Simkova a Stepanek, 2012)

2.3.1 HILL CLIMB

Metoda největšího gradientu je inspirována reálnou situací. Je-li zapotřebí dosáhnout vrcholu kopce je nejrychlejší postup po spádnici. Hill-Climbing je gradientní algoritmus určený k prohledávání stavového prostoru. U vstupního uzlu jsou sousední uzly ohodnoceny, analogicky takto algoritmus pokračuje u dalších uzlů a vždy vybírá nejlépe ohodnocený uzel. Algoritmus je ukončen v okamžiku, kdy je nalezen uzel s nejvyšším ohodnocením. (Borghoff et al., 2011)

2.3.1.1 Maximalizační algoritmus

Algoritmus řeší úlohy iterací k lokálnímu optimu. Jestliže je možno zajistit, aby lokální optimum bylo vždy také globálním optimem, lze získat algoritmus řešící tento problém. (Ahmed et al., 2012)

Tohoto se nepodařilo dosáhnout, stav je takový, že dochází k hledání pouze lokálního optima s předpokladem dostatečné blízkosti potřebného výsledku.

[27] viz Příloha 1 – Nástroj pro modelování rolí

Z tohoto důvodu je nutné zajistit větší množství počátečních bodů, aby se zvýšila pravděpodobnost nalezení dostatečně dobrého výsledku. (Mahdavi et al., 2003)

Jednotlivé kroky:

1. Volba počátečního stavu, je-li dostatečně dobrý – konec s výsledkem.
2. Pokud nebyl dosažen uspokojivý výsledek, dojde k prozkoumání okolí aktuálního stavu a výběru nejlepší varianty výsledku.
3. Není-li v okolí lepší stav nežli aktuální - konec bez výsledku.
4. Přechod do nového stavu z bodu 2.
5. Je-li aktuální stav dostatečně dobrý - konec, jinak přechod do bodu 2.

Volba počátečního stavu:

V tomto okamžiku se uplatňují předdefinované role a fixovaný atribut. Tato volba je akceptována tím že, dojde k označení rolí, které jsou považovány za neměnné (pro předdefinované role), nebo jednotlivé atributy (pro fixovaný atribut).

Využívá se několik strategií (heuristik) volby počátečního stavu, které se dále iteračně zlepšují. Je vydán první vyhovující výsledek. V případě nenalezení vyhovujícího výsledku, je navrácen nejlepší nalezený výsledek.

Heuristiky pro volbu počátečního stavu:

- Prázdné role (předpokládá se, že role vzniknou přidáváním atributů).
- Maximální role (předpokládá se, že role vzniknou odebíráním a změnou atributů).
- Role podle účtů.
- Náhodné role.

Předdefinované role:

Jsou vloženy do počátečního stavu spolu s ostatními rolemi, které budou dále upravovány. Jejich přidání je z důvodu ohodnocování celého stavu.

Fixovaný atribut:

Po zjištění hodnot, které jsou obsaženy v předdefinovaných rolích, jsou zbylé hodnoty vloženy do ostatních generovaných rolí v počátečním stavu. Toto je jediné místo, které může vést k chybě a to v případě, že není k dispozici dostatek rolí na umístění všech hodnot fixovaného atributu.

Okolí aktuálního stavu:

Za okolí aktuálního stavu je považován každý stav, do kterého se lze dostat změnou hodnoty jednoho atributu v jedné roli. Změna atributu znamená: přidání, odebrání celého atributu, nebo změna jeho hodnoty. Pro multi-value atributy je atribut přidáván s jednou hodnotou, odebírán jako celek a za změnu hodnoty je považováno přidání, nebo odebrání jedné z jeho hodnot.

Iterace stavů:

Není-li aktuální stav dostatečně dobrý (nepokrývá dostatečný počet účtů), jsou odzkoušeny všechny okolní stavy a z nich vybrán nejlepší, který je prohlášen za nový aktuální stav. Iterace je zastavena ve chvíli, kdy je nalezeno dostatečně dobré řešení, nebo již nejde aktuální stav dále zlepšovat (všechny okolní stavy jsou horší nežli aktuální).

Stavy jsou hodnoceny trojicí hodnot:

- Počet pokrytých účtů.
- Počet pokrytých atributů (i účet, který není zcela pokryt, může mít pokryty některé atributy).
- Počet blokujících atributů (představuje počet atributů, jejichž odstraněním nebo změnou by bylo možné roli použít na zkoumaný účet, výsledná hodnota vznikne jako suma přes všechny zkoumané účty).

2.3.1.2 Minimalizační algoritmus

Algoritmus vychází z verze pro hledání maximálního pokrytí a metody hledání řešení pomocí půlení intervalu spolu se znalostí hranic, za kterými se již řešení nenalézá.

Změna oproti metodě pro hledání maximálního pokrytí spočívá ve volbě počátečních stavů. Zatímco u hledání maximálního pokrytí startuje více horolezců při stejném počtu rolí, při hledání minimálního počtu rolí se klade důraz především na rychlost hledání a u každé skupiny rolí startuje pouze jeden horolezec.

Jednotliví horolezci postupují stejně jako v metodě pro hledání maximálního pokrytí účtů. Po dokončení běhu horolezce dojde k vyhodnocení, zda je počet rolí, se kterými byl spuštěn, dostatečný pro požadované pokrytí účtů. Za pomoci této informace se určí počet rolí, pro které bude spuštěn další horolezec. Při běhu horolezce také probíhá zlepšování jeho počátečního stavu, aby pokud možno nejlépe odpovídal pokrývaným účtům. Tohoto

lepšího stavu je využito při tvorbě počátečního stavu pro dalšího horolezce. Dosáhne se tak jeho zrychlení, jelikož již jeho počáteční stav obsahuje část jeho práce.

Volba počátečního stavu:

Dolním odhadem je minimální počet rolí nutný ke splnění podmínek a je to vyšší z těchto dvou hodnot – minimální počet rolí schopný pokrýt případný fixovaný atribut, nebo počet hodnot v XX% domény atributu s největší doménou, kde XX je velikost pokrytí zadané při spouštění algoritmu. Za horní odhad je brán počet agregovaných účtů, které je nutno pokrýt.

U prvního horolezce je nejprve nutné určit počet rolí u prvního horolezce. Zvolena je spodní mez, protože je rychleji vyhodnocena. První horolezec je spuštěn s první heuristikou, tedy s prázdnými rolemi. Je-li zadání fixovaný atribut, pak jsou jeho hodnoty rozloženy mezi generované role.

Následující horolezci již využívají rolí vygenerovaných předešlým horolezcem a upravených dle nového počtu testovaných rolí. Role jsou upravovány buď doplněním prázdnými rolemi, anebo odebráním nejméně využívaných rolí. Při odebírání rolí se však neodebírají předdefinované role nebo role nesoucí nějakou z hodnot fixovaného atributu. Tímto je zaručeno, že fixovaný atribut bude i nadále pokryt.

Iterace stavů:

Jedná se o metodu půlení intervalů, nový počet rolí je vybrán v půlce intervalu. Když dojde k střetnutí mezí intervalu, je určena výsledná hodnota. Výpočet na počátku algoritmu je mírně upraven. Počáteční horní mez je totiž příliš vysoká a vedla by k testování zbytečně velikého počtu rolí. Z tohoto důvodu se z počátku zvyšuje počet rolí podle poměru počet pokrytých účtů z minulého horolezce k počtu účtů požadovaných k akceptování řešení. Nedá-li se poměr určit, protože poslední horolezec nepokryl žádný účet, je počet rolí pro dalšího horolezce zvýšen na dvojnásobek oproti původní hodnotě.

Průběh algoritmu:

1. Určení počáteční dolní a horní meze.
2. Testování dolní meze pomocí testu s prázdnými rolemi.
3. Je-li pokrytí dostatečné, konec (dojde k vygenerování výstupu).
4. Zvýšení počtu rolí buď v poměru pokrytých a požadovaných účtů, nebo na dvojnásobek.

5. Testování nového počtu rolí pomocí upravených rolí z minulého testu.
6. Je-li pokrytí dostatečné, nastavení horní meze na aktuální počet rolí, jinak nastavení dolní meze na aktuální počet rolí.
7. Je-li horní mez větší než dolní mez pouze o jedna, ukončení (výstup s rolemi pro horní mez počtu rolí).
8. Určení nového počtu rolí v závislosti na mezích.
9. Návrat k bodu 5.

2.3.2 Genetický algoritmus

Genetický algoritmus je inspirovaný přirozenou evolucí. Aby našel dobré řešení, simuluje populaci možných řešení a pamatuje si nejlepší řešení, co se kdy objevilo. Toto nejlepší řešení vrátí jako výsledek po skončení běhu algoritmu.

Samotná simulace se skládá z množiny jedinců pevné velikosti. Každý jedinec reprezentuje sadu rolí, možné řešení problému. Simulace se dělí na generace, v každé generaci se vyberou vhodní jedinci a ostatní se zahodí. Z vhodných jedinců se různými operacemi vytvoří noví jedinci, jimiž je populace doplněna na původní velikost. Algoritmu skončí, pokud se po určitém počtu generací neobjevilo žádné lepší řešení.

Ohodnocující funkce:

Zjištění, kolik účtů řešení pokrývá, je náročná operace a zabírá drtivou většinu času při běhu algoritmu. Pro správný běh genetických algoritmů by měla být taková „ohodnocující funkce“, která se často používá i v ostatních algoritmech, velice rychlá. Přestože bylo snahou tuto funkci maximálně vylepšit, byla simulace z důvodu časové náročnosti ohodnocující funkce značně limitována. Přesto bylo nalezeno vhodné řešení, které umožňuje najít počet pokrytých účtů rychleji, jestliže se od posledního hledání pokrytí změnila jen jedna role.

2.3.2.1 Algoritmus v krocích

1. Naplnění části populace náhodnými jedinci odvozenými od agregovaných účtů.
2. Generování – vyplnění volného místa v populaci novými jedinci vytvořených ze stávajících.
3. Selekce – výběr vhodných jedinců, kteří zůstanou do další generace, zbytek bude odstraněn.

4. Pokud se za určitého počtu posledních generací neobjevilo žádné lepší řešení, konec. Vrácení nejlepšího řešení, co se během evoluce objevilo. Jinak navrácení do bodu 2.

2.3.2.2 Počáteční plnění populace

Místo zvolení úplně náhodného řešení jsou počáteční jedinci sestaveni z rolí ekvivalentních náhodným agregovaným účtům. Tyto účty jsou tedy přímo pokryty jednou odpovídající rolí, což by mělo řešení urychlit. Při maximalizačním problému je počet rolí na jedince zadán uživatelem před spuštěním a je fixní po celou dobu algoritmu. Počáteční řešení tedy odpovídá náhodně vybraným agregovaným účtům. Při minimalizačním problému rolí je každému řešení přiřazena pouze jedna takováto role, což umožní pozdější změny v rolích. Počet počátečních řešení je stejný, jako je počet jedinců, kteří zůstanou po selekci v každé generaci.

2.3.2.3 Generování nových jedinců

Pro doplnění populace jsou vytvářeni noví jedinci odvozením od stávajících. Jsou implementovány tři různé metody: mutace, spojení a jednoduchý horolezec.

Mutace

Základní a nejpoužívanější metoda, od jednoho jedince je odvozen druhý. Je provedena jedna z následujících operací:

Přidání nebo odebrání celé role

První operace mutace je možná pouze při minimalizačním problému. Je to přidání nebo odebrání celé role. Pokud toto řešení ještě nesplnilo minimální pokrytí, je větší šance pro přidání role. Je-li už minimální pokrytí splněno, nastává větší šance pro odebrání role.

Nahrazení jedné role jinou náhodně generovanou

Nahrazení probíhá přidáním nebo odebráním atributu z jedné náhodně vybrané role. Je potřeba mít na paměti, že na rozdíl od účtů ne všechny atributy v roli musí být definovány. Poměr přidání nebo odebrání je stejný. Pokud ale v roli zbývá již poslední atribut, je určitě jeden atribut přidán. Naopak, pokud jsou všechny atributy přítomny, jeden se odebere. Tato operace tedy nenechá roli prázdnou.

Výměna hodnoty atributu

Výměnu hodnoty stávajícího náhodně vybraného atributu z náhodně vybrané role za jinou náhodnou hodnotu lze provést za předpokladu, že to není *multi-value* atribut. Pokud se jedná o *multi-value* atribut, lze odebrat hodnotu, přidat hodnotu, nebo obojí najednou, což znamená vyměnit jednu hodnotu za jinou. Tato operace neodebere poslední hodnotu.

Spojení

Operace se také nazývá křížení. Spolu s mutací je to jedna ze základních akcí v genetickém algoritmu. Testy ukázaly, že používání této metody vede k horším výsledkům. Není možné použít optimalizaci při zjišťování pokrytí, protože se mění celé řešení, nikoli jen jedna role. Navíc časté používání spojení může vést k velké podobnosti všech řešení a tím pádem ke ztrátě různorodosti. Uživatel si však tuto možnost může vyzkoušet.

Samotné základní provedení je jednoduché. Dvě stávající řešení jako řetěz atributů poskládaný z přítomných rolí se spojí na obou koncích do kruhu. Z prvního řešení se použije náhodná část a zbytek se doplní z druhého řešení.

Spojení funguje pro hledání maximálního pokrytí, kde velikosti řešení jsou stejné, je stejný počet rolí. Při hledání minimálního počtu rolí je problém v tom, že jsou řešení různě velká. Proto se v řešení s větším počtem rolí ignoruje tolik rolí, o kolik je toto řešení větší než to druhé. Určitá část většího řešení tak není vůbec použita. Aby se tato část neztratila, je z ní vybrán náhodný úsek rolí a přidán do nového jedince.

Jednoduchý horolezec

V genetických algoritmech se ukázalo, že je někdy dobré proložit náhodné hledání cíleným zlepšováním řešení. Důvodem je využití šance posunout populaci správným směrem. Tato metoda náhodně zvolí atribut v řešení a systematicky zkouší měnit jeho hodnoty s cílem zlepšit řešení. Až vyčerpá všechny hodnoty, postoupí k dalšímu atributu. Má daný počet kroků a počet hodnot, co může vyzkoušet. Problémem je, že při každém kroku musí zjistit pokrytí, tedy doba běhu této metody je alespoň tolikrát delší než mutace, kolik kroků je provedeno. Proto se jako křížení běžně nepoužívá. Pokud nenalezne žádné lepší řešení, nevytvoří žádného nového jedince.

2.3.2.4 Selekce

Vybere určitý počet nejlepších řešení a zbytek smaže. Spolu s nejlepšími řešeními však ponechá ještě malý počet náhodně vybraných řešení, aby nebyly ztraceny všechny možnosti evoluce.

Někdy se stane, že populace není doplněna na úplné maximum (např. jednoduchý horolezec nemusí najít lepší řešení, nebo při selekci je ponecháno méně jedinců kvůli většímu počtu vhodných řešení). Takovéto výkyvy jsou však nepodstatné, jsou vyrovnány v další generaci tím způsobem, že se při plánování počtu generovaných jedinců metodou mutace přidá počet chybějících jedinců.

Předdefinované role:

Pokud byly zadány předdefinované role, přidají se k řešení vždy jen na zjišťování pokrytí a pak se zas odeberou, aby nezabíraly zbytečně místo. Jinak vliv na běh algoritmu nemají.

Fixovaný atribut:

Každé řešení v populaci splňuje podmínku fixovaného atributu. Pokud náhodou některá metoda tuto podmínku narušuje, upraví se řešení tak, aby se to napravilo. To často vede k tomu, že v dané metodě je změněna více než jedna role a pak se nedá použít pro daný krok. Jinak se fixovaný atribut neprojevuje.

2.3.3 SAT algoritmus

SAT (Boolean SATisfiability problem) je rozhodovací NP-úplný problém, jehož definice je následující: Je dána formule na logických proměnných v konjunktivní normální formě. Existuje (pravdivostní) ohodnocení proměnných, pro které má formule hodnotu 1. Jinými slovy řešíme problém pravdivosti uzavřené existenčně kvantifikované booleovské formule.

Příklad:

(2-28)

$$\exists\, x1\, x2\, x3\, x4\, (x1 \vee \neg\, x2 \vee \neg\, x3) \wedge (x1 \vee x3 \vee x4$$

Řešení:

Formule je splněna když: $x1,\ x2,\ x3,\ x4 = TRUE$

2.3.3.1 Převod problému na SAT

V této kapitole je nejprve popsán problém stylem shora-dolů, kdy je popis složitějšího objektu dekomponován a sestaven z popisů objektů jednodušších. Vzhledem k rozsahu převodu problému na SAT algoritmus jsou zde zmíněny pouze základní definice. Při převodu problému na SAT je celková formule pokrytí všech zadaných agregovaných účtů dekomponována na jednodušší pod formule, které popisují pokrytí objektů jednodušších. Později dochází k převodu formule do konjunktivní normální formy. (Simkova, 2013)

Základní definice a označení:

$agregAcc$ je agregovaný účet, kdy $AGREG\_ACCOUNTS$ je označení pro množinu všech zadaných agregovaných účtů, přičemž $AGREG\_ACCOUNTS(i)$ představuje i-tý agregovaný účet množiny všech zadaných agregovaných účtů. $attrib$ je atribut, který má svůj název $NAME(attrib)$, typ $TYPE(attrib)$. Množina všech atributů agregovaného účtu $agregAcc$ je označena jako $ATTRIBS(agregAcc)$. Množina všech zadaných atributů k pokrytí $ATTRIBS\_TO\_COVER$ nesoucí atribbut k pokrytí $attribToCover$ označený názvem atributu $NAME(attribToCover)$. Roli představuje $role$ a množina všech rolí, které má algoritmus vytvořit je $ROLES$, přičemž k-tá role, která má algoritmus vytvořit je $ROLES(k)$. Dále je množina všech atributů role označena jako $ATTRIBS(role)$, $AttribValue$ je hodnota atributu, $VALUES(attrib)$ množina všech hodnot atributu $attrib$, logická proměnná $logVariable$ a hodnota logické proměnné $BOOL\_VALUE(logVariable)$.

Základní axiomy

(2.2.23)

$$i \in 1..|AGREG\_ACCOUNTS|$$

$$agregAcc \in AGREG\_ACCOUNTS$$

$$attrib : \exists\, agregAcc, attrib \in ATTRIBS\,(agregAcc)$$

$$attribToCover \in ATTRIBS\_TO\_COVER$$

$$role \in ROLES$$

$$attribValue : \exists agregAcc, \exists attrib, \exists attribName, NAME(attrib) = attribName, attribValue \in$$
$$\in VALUES((ATTRIBS(agregAcc))(attribName))$$

$$TYPE(attrib) \in \{highest\_Value, priority, union\}$$

$$\forall attrib, TYPE(attrib) \in \{highest\_Value, priority\} : |VALUES(attrib)| \leq 1$$

$$\forall attribToCover \in ATTRIBS\_TO\_COVER, \forall agregAcc \in AGREG\_ACCOUNTS:$$
$$\exists attrib \in ATTRIBS(agregAccount) \wedge NAME(attrib) = NAME(attribToCover)$$

$$attrib \in ATTRIBS(agregAcc) : (ATTRIBS(agregAcc))(NAME(attrib)) = attrib$$
$$attrib \notin ATTRIBS(agregAcc) : (ATTRIBS(agregAcc))(NAME(attrib)) = \varnothing$$

$$attrib \in ATTRIBS(role) : (ATTRIBS(role))(NAME(attrib)) = attrib$$

$$attrib \notin ATTRIBS(role) : (ATTRIBS(role))(NAME(attrib)) = \varnothing$$

$$attribToCover \in ATTRIBS\_TO\_COVER : ATTRIBS\_TO\_COVER(NAME(attribToCover)) = attribToCover$$

$$attribToCover \notin ATTRIBS\_TO\_COVER : ATTRIBS\_TO\_COVER(NAME(attribToCover)) = \varnothing$$

$$BOOL\_VALUE(\log Variable) \in \{TRUE, FALSE\}$$

Doména atributu

Doména atributu DOM s daným názvem $domainName$ je definována jako sjednocení všech množin hodnot všech atributů daného názvu všech agregovaných vstupních účtů, přičemž doména atributu, jehož název je $domain\_name$ představuje $DOM(domain\_name)$.

Axiomy domény atributu

(2.2.24)

$$domainName : \exists attribToCover, NAME(attribToCover) = domainName$$

$$\forall agregAcc \in AGREG\_ACCOUNTS, \forall attrib \in (ATTRIBS(agregAcc))(domainName):$$

$$DOM(domainName) = \cup VALUES(attrib)$$

Základní proměnné

Daná základní proměnná $BASE\_VARIABLES(role)$ představující množinu všech logických základních proměnných role $role$ je vždy asociována s jistou rolí, atributem k

pokrytí $attribToCoverName$ a jistou hodnotou z domény tohoto atributu k pokrytí. Tato základní proměnná nabývá hodnoty $TRUE$ právě tehdy, když daná role obsahuje v daném atributu danou hodnotu, přičemž množina všech logických základních proměnných definovaných pro $role$ a $attribToCoverName$ je označena jako $BASE\_VARIABLES\ (role, attribToCoverName)$ a základní proměnná pro danou roli, daný název atributu k pokrytí a danou hodnotu atributu k pokrytí dané role je $BASE\_VARIABLES\ (role, attribToCoverName, attribValue)$.

Axiomy základních proměnných

(2.2.25)

$$attribToCo\ verName\ : \exists attribToCo\ ver\ ,\ NAME\ (attribToCo\ ver\ ) = attribToCo\ verName$$

$$\forall role : BASE\_VARIABLES(role) = \bigcup_{attribToCoverName} BASE\_VARIABLES(role, attribToCoverName)$$

$$\forall role, \forall attribToCoverName : BASE\_VARIABLES(role, attribToCoverName) =$$

$$= \bigcup_{attribValue \in DOM\ (attribToCoverName)} BASE\_VARIABLES(role, attribToCoverName, attribValue)$$

$$\forall role, \forall attribToCoverName, \forall attribValue :$$

$$BOOL\_VALUE(BASE\_VARIABLE(role, attribToCoverName, attribValue)) \in \{TRUE\ , FALSE\}$$

$$BOOL\_VALUE(BASE\_VARIABLE(role, attribToCoverName, attribValue)) = TRUE \Leftrightarrow$$

$$\Leftrightarrow attribValue \in VALUES((ATTRIBS(role))(attribToCoverName))$$

Pravidla základních proměnných

Pro typy atributů s jednoduchou hodnotou platí, že u proměnných (základní proměnná $baseVariable$), které jsou asociované s danou rolí a daným atributem, existuje nejvýše jedna taková, která má hodnotu $TRUE$. Pro atributy typu multi-value toto omezení neplatí.

Axiomy pravidel základních proměnných

(2.2.26)

$$baseVariable \in \bigcup_{\forall role \in ROLES} BASE\_VARIABLES\ (role)$$

$$\forall role, \forall attribToCoverName: \left\| \left\{ \begin{array}{l} baseVariable : baseVariable \in BASE\_VARIABLES(role, \\ attribToCoverName), TYPE(ATTRIBS\_TO\_COVER(\\ attributeToCoverName)) \in \{highest\_value, priority\}, \\ BOOL\_VALUE(baseVariable) = TRUE \end{array} \right\} \right\| \leq 1$$

2.3.3.2 Formulace problémů při převodu na SAT

Problém č. 1

Rozhodnout, zda je daná množina agregovaných účtů $AGREG\_ACCOUNTS$ pokryta daným počtem rolí v dané množině atributů k pokrytí.

Množina všech zadaných agregovaných účtů $AGREG\_ACCOUNTS$ je pokryta právě tehdy, když je pokryt daným počtem rolí $ROLES$ každý agregovaný účet této množiny. Tedy množinou všech rolí $ROLES$, které má algoritmus vytvořit, přičemž množinu všech zadaných atributů k pokrytí tvoří $ATTRIBS\_TO\_COVER$. Množinu všech logických proměnných, reprezentujících existenci pokrytí $AGGREG\_ACCOUNTS$ reprezentuje ACC_COV a i-tá, existence pokrytí účtu reprezentující proměnnou formuje $ACC\_COV(i)$.

Axiomy

(2.2.27)

$$|ACC\_COV| = n$$

$$i \in 1..n$$

$$\forall i : ACC\_COV(i) \in ACC\_COV$$

$$BI : AGREG\_ACCOUNTS \leftrightarrow ACC\_COV$$

$$\forall i : BOOL\_VALUE(ACC\_COV(i)) \in \{TRUE, FALSE\}$$

$$\forall i : BOOL\_VALUE(ACC\_COV(i)) = TRUE \Leftrightarrow ACC\_COV(i) = BI(agregAcc), agregAcc$$

Formule

(2.2.28)

$$F\_COV = ACC\_COV(1) \wedge ACC\_COV(2) \wedge ACC\_COV(3) \wedge ..$$

$$\wedge ACC\_COV(n)$$

Problém č. 2

Rozhodnout, zda je daný agregovaný účet in_aggAcc pokryt daným počtem rolí v dané množině atributů k pokrytí.

Účel je tedy pokryt ($ATTRIBS\_TO\_COVER$) právě tehdy, když je pokryt každý jeho atribut, jehož název je obsažen v množině názvů atributů k pokrytí, přičemž $ATTRIBS\_COV$ reprezentuje množinu všech logických proměnných reprezentujících existenci pokrytí všech atributů k pokrytí zadaného agregovaného účtu a $ATTRIBS\_COV_{(i)}$ představuje i-tou, existenci pokrytí atributu účtu reprezentující proměnná.

Popis pokrytí atributu lze rozdělit do dvou případů podle typu atributu. Jednodušší je případ, kdy je atribut typu highest-value nebo typu priority. Atribut tohoto typu obsahuje v rámci každého účtu pouze jednu hodnotu. Složitější je situace u atributu typu multi-value, kde může být hodnot více.

Axiomy

(2.2.29)

$$|ATTRIBS\_COV| = m$$

$$i \in 1..m$$

$$\forall i : ATTRIBS\_COV(i) \in ATTRIBS\_COV$$

$$ATTRIBS\_MUST\_COV = \left\{ \begin{array}{l} \forall attrib : ATTRIBS(agregAcc), NAME(attrib) \in \\ \in \bigcup_{\forall attribToCover} NAME(attribToCover) \end{array} \right\}$$

$$BI : ATTRIBS\_MUST\_COV \leftrightarrow ATTRIBS\_COV$$

$$\forall i : BOOL\_VALUE(ATTRIBS\_COV(i)) \in \{TRUE, FALSE\}$$

$$\forall i : BOOL\_VALUE(ATTRIBS\_COV(i)) = TRUE \Leftrightarrow ATTRIBS\_COV(i) =$$

$$BI(attrib), attrib$$

Formule

(2.2.30)

$$F\_ACC\_COV = ATTRIBS\_COV(1) \wedge ATTRIBS\_COV(2) \wedge .. \wedge ATTRIBS\_COV(m)$$

Problém č. 3

Rozhodnout, zda je daný atribut in_attrib jednoduchého typu daného agregovaného účtu pokryt daným počtem rolí $ROLES$ v dané množině atributů k pokrytí $ATTRIBS\_TO\_COVER$. $ROLES$ představuje množinu všech rolí, které má algoritmus vytvořit, přičemž $ROLES\_COV$ tvoří množinu všech logických proměnných, reprezentujících existenci pokrytí zadaného atributu množinou $ROLES$. I v tomto

problému existuje i-tá logická proměnná $ROLES\_COV_{(i)}$, která reprezentuje existenci pokrytí zadaného atributu specifickou rolí.

Daný atribut jednoduchého typu daného agregovaného účtu je pokryt právě tehdy, když je pokryt alespoň jednou rolí tohoto účtu.

Axiomy

(2.2.31)

$$|ROLES\_COV| = r$$

$$i \in 1..r$$

$$\forall i : ROLES\_COV(i) \in ROLES\_COV$$

$$BI : ROLES \leftrightarrow ROLES\_COV$$

$$\forall i : BOOL\_VALUE(ROLES\_COV(i)) \in \{TRUE, FALSE\}$$

$$\forall i : BOOL\_VALUE(ROLES\_COV(i)) = TRUE \Leftrightarrow ROLES\_COV(i) = BI(role), role$$

Formule

(2.2.32)

$$F\_SIMPLE\_ATTRIB\_COV = ROLES\_COV(1) \vee ROLES\_COV(2) \vee ..$$

$$\vee\ ROLES\_COV(r)$$

Problém č. 4

Rozhodnout, zda je daný atribut in_attrib typu multi-value daného agregovaného účtu pokryt $ATTRIBS\_TO\_COVER$ daným počtem rolí $ROLES$ v dané množině atributů k pokrytí, přičemž $VALUES\_COV$ je množina všech logických proměnných reprezentujících existenci pokrytí všech hodnot z $VALUES(attrib)$ a $VALUES\_COV_{(i)}$ formuje i-tou logickou proměnnou, která přestavuje existenci pokrytí určité hodnoty zadaného atributu.

Zadaný atribut typu multi-value je pokryt právě tehdy, když je pokryta daným počtem rolí každá jeho hodnota. Popis pokrytí určité hodnoty atributu typu muti-value je týž jako popis pokrytí jednoduchého atributu s toutéž hodnotou.

Axiomy

(2.2.33)

$$|VALUES\_COV| = k$$

$$i \in 1..k$$

$$\forall i : VALUES\_COV(i) \in VALUES\_COV$$

$$BI : VALUES(attrib) \leftrightarrow VALUES\_COV$$

$$\forall i : BOOL\_VALUE(VALUES\_COV(i)) \in \{TRUE, FALSE\}$$

$$\forall i : BOOL\_VALUE(VALUES\_COV(i)) = TRUE \Leftrightarrow$$

$$\Leftrightarrow VALUES\_COV(i) = BI(attribValue), attribValue$$

Formule

(2.2.34)

$$F\_UNION\_ATTRIB\_COV = VALUES\_COV(1) \wedge VALUES\_COV(2) \wedge ..$$

$$\wedge VALUES\_COV(k)$$

Problém č. 5

Rozhodnout, zda daná role in_role pokrývá danou hodnotu $in\_attribValue$ daného atributu in_attrib daného agregovaného účtu $in\_agregAcc$. $ROLE\_ADEPTS$ představuje množinu logických proměnných reprezentujících existenci rolí jako adeptů na pokrytí agregovaných účtů. $ROLE\_ADEPT$ $(role, agregAcc)$ je logická proměnná reprezentující existenci $role$ jako adepta na pokrytí $agregAcc$.

Zadaná hodnota je pokryta právě tehdy, když je zadaná role adept na pokrytí zadaného agregovaného účtu a zadaná role obsahuje v daném atributu danou hodnotu.

Axiomy

(2.2.35)

$$ROLE\_ADEPTS = \bigcup_{\forall role} \bigcup_{\forall agregAcc} ROLE\_ADEPT(role, agregAcc)$$

$$\forall role, \forall agregAcc : BOOL\_VALUE(ROLE\_ADEPT(role, agregAcc)) \in \{TRUE, FALSE\}$$

$$BOOL\_VALUE(ROLE\_ADEPT(role, agregAcc)) = TRUE \Leftrightarrow role$$

Formule

(2.2.36)

$$F\_VALUE\_COV = BOOL\_VALUE(ROLE\_ADEPT(in\_role, in\_agregAcc)) \wedge$$

$$BOOL\_VALUE(BASE\_VARIABLE(in\_role, NAME(in\_attrib), in\_attribValue)$$

Problém č. 6

Rozhodnout, zda je daná role in_role adeptem na pokrytí daného agregovaného účtu $in\_agregAcc$ v dané množině atributů k pokrytí $ATTRIBS\_TO\_COVER$, kdy $ADEPT\_ATTRIB\_COV$ představuje množinu všech logických proměnných

reprezentujících existence adeptů na pokrytí všech atributů $in\_agregAcc$, které mají být porkyty. $ADEPT\_ATTRIB\_COV_{(i)}$ formuje i-tou logickou proměnnou množiny $ADEPT\_ATTRIB\_COV$.

Daná role je adeptem na pokrytí daného agregovaného účtu v dané množině atributů k pokrytí právě tehdy, když je adeptem na pokrytí všech atributů k pokrytí tohoto účtu.

Axiomy

(2.2.37)

$$|ADEPT\_ATTRIB\_COV| = m$$

$$i \in 1..m$$

$$\forall i : ADEPT\_ATTRIB\_COV(i) \in ADEPT\_ATTRIB\_COV$$

$$ATTRIBS\_MUST\_COV = \left\{ \begin{array}{l} \forall attrib : ATTRIBS(agregAcc), NAME(attrib) \in \\ \in \bigcup_{\forall attribToCover} NAME(attribToCover) \end{array} \right\}$$

$$BI : ATTRIBS\_MUST\_COV \leftrightarrow ADEPT\_ATTRIB\_COV$$

$$\forall i : BOOL\_VALUE(ADEPT\_ATTRIB\_COV(i)) \in \{TRUE, FALSE\}$$

$$\forall i : BOOL\_VALUE(ADEPT\_ATTRIB\_COV(i)) = TRUE \Leftrightarrow$$
$$\Leftrightarrow ADEPT\_ATTRIB\_COV(i) = BI(attrib),\ in\_role$$

Formule

(2.2.38)

$$F\_ROLE\_ADEPT = ADEPT\_ATTRIB\_COV(1) \wedge ADEPT\_ATTRIB\_COV(2) \wedge$$

$$.. \wedge ADEPT\_ATTRIB\_COV(m)$$

Problém č. 7

Rozhodnout, zda je daná role in_role adeptem na pokrytí daného atributu in_attrib daného agregovaného účtu $in\_agregAcc$, přičemž $FORBIDDEN\_VARS$ představuje množinu základních proměnných, které nesmí nabývat hodnoty $TRUE$, vymezení této množiny závisí na typu vstupního atributu. $FORBIDDEN\_VARS_{(i)}$ je i-tá proměnná množiny $FORBIDDEN\_VARS$.

Rozhodnutí záleží na typu atributu, který se má pokrýt danou rolí. Pro atribut typu *highestValue* nesmí základní proměnné dané role, které jsou asocovány s hodnotami atributu stejného názvu jako má daný atribut, nabývat hodnoty *TRUE* právě tehdy, když reprezentují celočíselnou hodnotu, která je vyšší než celočíselná hodnota daného atributu u daného agregovaného účtu. Jinými slovy celočíselná hodnota atributu musí být u zadané role, která se bude generovat, nižší nebo stejně velká, jako je hodnota atributu téhož jména u zadaného účtu. Pokud tomu tak nebude, zadaná role nebude zadaný účet v tomto atributu pokrývat. U atributu typu *priority* se testuje na shodu hodnoty zadané role a atributu ve stejnojmenném atributu, který se má pokrýt. U atributu typu *union* se testuje, zda hodnoty zadané role budou tvořit podmnožinu hodnot stejnojmenného atributu zadaného účtu.

Axiomy

(2.2.39)

$$|FORBIDDEN\_VARS| = f$$

$$i \in 1..f$$

$$\forall i : FORBIDDEN\_VARS(i) \in FORBIDDEN\_VARS$$

$$\text{pro } TYPE(in\_attrib) = highest\_value :$$

$$\forall attrib, VALUES(attrib) = \emptyset : INT\_VALUE(VALUES(attrib)) = 0$$

$$FORBIDDEN\_VARS = \left\{ \begin{array}{l} \forall BASE\_VARIABLE(in\_role, NAME(in\_attrib), attribValue): \\ INT\_VALUE(attribValue) > INT\_VALUE(VALUES(in\_attrib)) \end{array} \right\}$$

$$\text{pro } TYPE(in\_attrib) = priority :$$

$$FORBIDDEN\_VARS = \left\{ \begin{array}{l} \forall BASE\_VARIABLE(in\_role, NAME(in\_attrib), attribValue) : \\ attribValue \neq VALUES(in\_attrib)) \end{array} \right\}$$

$$\text{pro } TYPE(in\_attrib) = union :$$

$$FORBIDDEN\_VARS = \left\{ \begin{array}{l} \forall BASE\_VARIABLE(in\_role, NAME(in\_attrib), attribValue): \\ attribValue \notin VALUES(in\_attrib)) \end{array} \right\}$$

Formule

(2.2.40)

$$F\_ADEPT\_ATTRIB = \neg FORBIDDEN\_VARS(1) \wedge \neg FORBIDDEN\_VARS(2) \wedge ...$$
$$\wedge \neg FORBIDDEN\_VARS(f)$$

2.3.3.3 Převod rovnice do konjunktivní normální formy (CNF)

Jelikož použitý SAT řešič, neumožňuje zadat libovolnou formuli, ale formuli ve formě CNF, je nutné převést dříve vypracovaný převod do tohoto tvaru. Postup je založen na použití ***logických implikací***.

Pro každou formuli, která je instancí některého z typů formulí definovaných v předchozí části, je vytvořena nová proměnná, která bude poté implikovat pravou stranu formule. Poté se tato nově vzniklá implikace převede na CNF. Klauzule, které vzniknou převodem na CNF, se potom dají přímo do řešiče.

Příklad:

(2.2.41)

$$F\_ACC\_COV = ATTRIBS\_COV(1) \wedge ATTRIBS\_COV(2) \wedge .. \wedge ATTRIBS\_COV(m)$$

1. Nejprve se vytvoří nová logická proměnná F_ACC_VAR
2. Vytvoří se implikace:

(2.2.42)

$$F\_ACC\_VAR \Rightarrow ATTRIBS\_COV(1) \wedge ATTRIBS\_COV(2) \wedge ... \wedge ATTRIBS\_COV(m)$$

3. Převodem na CNF vzniknou nové klauzule, které se dají již přímo do řešiče:

(2.2.43)

$$(\neg F\_ACC\_VAR \vee ATTRIBS\_COV(1)) \wedge (\neg F\_ACC\_VAR \vee ATTRIBS\_COV(2)) \wedge ...$$
$$... \wedge (\neg F\_ACC\_VAR \vee ATTRIBS\_COV(m))$$

4. Pokud je hodnota proměnné F_ACC_COV rovna $TRUE$, potom je asociovaný účet pokryt.

Hierarchickým přístupem se takto vytvoří celá hlavní formule a místo původních pod formulí se používají tímto způsobem nově vzniklé proměnné. Každý objekt dostane svou logickou proměnnou, jejíž hodnota bude představovat jakýsi stav tohoto objektu, většinou to bude stav pokrytí daného objektu, viz příklad nahoře.

Implikace vs. Ekvivalence

Použití ekvivalence místo implikace při zpracovávání jednotlivých pod-formulí by bylo nejen přirozené, ale i přesnější. Použití implikace totiž připouští případ, kdy je hodnota jisté proměnné na levé straně implikace $FALSE$, zatímco její pravá strana má hodnotu $TRUE$. To např. znamená, že ačkoli jsou pokryty všechny atributy jistého účtu, účet sám není pokryt, což je v rozporu s původní definicí a popisem problému.

V praxi se však ukazuje nevýhodnost použití ekvivalencí. Jednou příčinou je to, že počet vytvořených klauzulí by při použití ekvivalencí značně narostl – průměrně zhruba na dvojnásobek, čímž by narostla spotřeba paměti, která zvláště při výpočtu větších instancí hraje velikou roli. Druhá příčina, která částečně souvisí s první, je znatelné prodloužení výpočtu, což se zvláště u větších instancí znatelně projeví.

I když je použití implikací v rozporu s původní definicí a popisem problému, nepřináší tento přístup potíže. Řešič se bude snažit ohodnotit všechny proměnné tak, aby byly všechny klauzule (disjunkce) splněny. Jelikož se proměnné agregovaných účtů nachází v jednotkové disjunkci (má pouze 1 člen), musí řešič přiřadit těmto proměnným hodnotu $TRUE$. Tato hodnota, ale již implikuje splnění pravé strany implikace, tedy klade podmínky na hodnoty objektů v nižších částech hierarchie, nikoliv naopak, takže celá ekvivalence se zde neuplatní.

Rozepsání agregovaných účtů

Vzhledem k tomu, jak je implementován maximalizační algoritmus, který bude popsán v následující kapitole, je třeba do výsledné formule reálné účty také zakomponovat.

Tímto zakomponováním je myšlena schopnost mít každý reálný účet asociovaný s danou logickou proměnnou, jejíž hodnota je $TRUE$ nebo $FALSE$ (účet pokryt nebo nepokryt). Hodnoty těchto proměnných budou jistě ekvivalentní hodnotám proměnných příslušných agregovaných účtů. Proto se do výsledné formule přidávají ještě CNF transformace těchto ekvivalencí.

2.3.3.4 Maximalizační algoritmus

Úkolem maximalizačního algoritmu je najít největší podmnožinu účtů takovou, která bude pokryta daným počtem vygenerovaných rolí. Pokud se do řešiče umístí transformovaný problém popsaný v předchozí kapitole, řešič oznámí, zda je možné pokrýt všechny účty vstupní množiny nebo nikoliv, informace o nejvyšším dosaženém pokrytí účtů není k dispozici. Přístupů, jak řešit tento problém, je několik.

MAXSAT

Je rozhodovací NP-těžký problém, jehož definice je následující: Je dána formule F na n logických proměnných v konjunktivní normální formě. Jaký je maximální možný počet splnitelných klauzulí této formule?

Nejznámějším rozšířením MAXSATu je Weighted MAXSAT (vážený MAXSAT), který se snaží najít maximální váhu celé formule, přičemž každá klauzule má přiřazenou váhu.

Specifikovat problém tak, aby odpovídal vstupnímu formátu řešiče pro Weighted MAXSAT je snadné. Každá implikace – jí odpovídající klauzule v CNF, dostane maximální možnou váhu, protože musí platit vždy. Jednotkové disjunkce obsahující pouze proměnné samotných agregovaných účtů dostanou váhu podle počtu jejich reálných účtů. V tom případě se ale nebudou do výsledné formule přidávat disjunkce pro každý reálný účet, ale pouze pro účty agregované.

Avšak praktické pokusy na větších instancích ukázaly praktickou nepoužitelnost tohoto přístupu, neboť spotřebovaný čas neúměrně roste.

Iterativní zvyšování počtu pokrytých účtů

Ideou tohoto přístupu je myšlenka, že pokus pokrýt pouze jistou podmnožinu nižší kardinality všech vstupních účtů je v běžném případě nesrovnatelně jednodušší s ohledem na spotřebu systémových prostředků – času a paměti, než pokus pokrýt celou tuto množinu. Navíc, nalezené řešení lze uchovat. Po nalezení řešení pro tuto podmnožinu lze celý algoritmus iterovat pro nalezení podmnožiny vyšší kardinality, než měla předchozí podmnožina. Algoritmus lze zastavit po dosažení předem definované hodnoty spotřebovaného výpočetního času nebo po nalezení řešení pro podmnožinu definované kardinality.

Pro každý reálný účet z množiny vstupních účtů nechť je definována logická proměnná, která nabývá hodnoty $TRUE$ právě tehdy, když je tento reálný účet pokryt danou sadou

rolí. Nechť je definována booleovská funkce, jejíž arita je rovna kardinalitě množiny vstupních účtů. Tato funkce nechť nabývá hodnoty $TRUE$ právě tehdy, když je počet jejích argumentů nabývajících hodnoty $TRUE$, rovný nebo větší než jisté číslo k. Za předpokladu, že argumenty této funkce budou proměnné reálných účtů, to znamená, že tato funkce nabývá hodnoty $TRUE$ právě tehdy, když je počet reálných účtů, pokrytých danou sadou rolí, roven nebo větší než k.

Přidání reálných účtů do výsledné formule

Jak je již zřejmé z popisu iterativního přístupu, je nutné do výsledné formule zakomponovat též proměnné reálných účtů. Za každou proměnnou reálného účtu se do formule přidá jednotková disjunkce, která bude obsahovat právě tuto proměnnou.

Dále je nutné přidat vazby na proměnné příslušných agregovaných účtů. Aktuální hodnoty proměnných reálných účtů budou ekvivalentní hodnotám proměnných příslušných účtů agregovaných. Proto se do výsledné formule přidají ještě CNF transformace těchto ekvivalencí.

Identifikace podmnožin účtů pomocí grafu

Myšlenka tohoto přístupu je založena na faktu, že graf ohodnocení booleovské funkce, která bude reprezentovat kladné ohodnocení daného minimálního počtu proměnných asociovaných s účty, lze vhodnými úpravami komprimovat až do té míry, že počet uzlů tohoto grafu bude výpočetně přiměřený, to znamená, že nebude exponenciální v počtu účtů.

Graf ohodnocení booleovské funkce

Graf ohodnocení booleovské funkce f s n proměnnými je binární strom $G(V, E)$ pro který platí následující.

Na množině proměnných funkce f je definováno uspořádání, kde hloubka stromu $h = n$ a existuje bijektivní zobrazení z množiny proměnných funkce f do množiny hladin stromu. Každý vrchol stromu představuje unikátní množinu proměnných f s jednoznačně přiřazenými hodnotami. Kořen stromu představuje situaci, kdy nebyla přiřazena hodnota do žádné proměnné. Listy definují situace, kdy má každá proměnná přiřazenu svou hodnotu a hodnota funkce je tudíž známá. Levý potomek daného vrcholu představuje situaci, kdy je proměnná příslušná hladině vrcholu ohodnocena na $FALSE$ a pravý

potomek daného vrcholu představuje situaci, kdy je proměnná příslušná hladině vrcholu ohodnocena na $TRUE$.

Jak již bylo uvedeno dříve, každý vrchol grafu je spojen s jistým počtem kladně ohodnocených proměnných reálných účtů. Každý vrchol se nachází v jednom ze tří možných stavů, a to $SATISFIED$, $UNSATISFIED$ a $INNER$. $SATISFIED$ určuje počet proměnných reálných účtů, ohodnocených na $TRUE$ a je roven nebo větší než k. $UNSATISFIED$ je stav, kdy součet počtu proměnných reálných účtů, ohodnocených na $TRUE$a počtu dosud neohodnocených proměnných reálných účtů, je menší než k. Pro poslední stav $INNER$ neplatí žádný obou předchozích stavů.

Pro využití grafu platí booleovská funkce $atLeastSatisfied(ACCOUNT\_VARS)$, definována na množině proměnných reálných účtů $ACCOUNT\_VARS$. Tato funkce má hodnotu $TRUE$ právě tehdy, když je počet proměnných reálných účtů roven nebo větší předem dané hodnotě k prezentující počet proměnných ohodnocených na $TRUE$. Vzhledem k významu proměnných to tedy znamená, že tato funkce má hodnotu $TRUE$ právě tehdy, když je počet reálných účtů pokrytých rolemi roven nebo větší než k. Aby mohl řešič řešit úlohu, zda existuje pokrytí minimálně k reálných účtů, je třeba k celkové formuli přidat ještě podmínku kladné hodnoty funkce $atLeastSatisfied(ACCOUNT\_VARS)$ s předem daným k. K tomu se právě využije graf ohodnocení.

Axiomy grafu ohodnocení booleovské funkce

(2.2.44)

$$\left|\{\forall accountVar \in ACCOUNT\_VARS : accountVar = TRUE\}\right| < k \Leftrightarrow$$

$$\Leftrightarrow atLeastSatisfied(ACCOUNT\_VARS) = FALSE$$

$$\left|\{\forall accountVar \in ACCOUNT\_VARS : accountVar = TRUE\}\right| \geq k \Leftrightarrow$$

$$\Leftrightarrow atLeastSatisfied(ACCOUNT\_VARS) = TRUE$$

Princip ohodnocení pomocí formule v CNF

Každý uzel $ROOT(G)$ grafu G dostane přidělenu svou logickou proměnnou $VERTEX\_VARIABLES$, jejíž kladná hodnota bude znamenat, že nastala situace příslušná tomuto uzlu. Pokud nastala situace příslušná tomuto uzlu, musela nastat situace příslušná předchůdci tohoto uzlu $PARENT(v)$, spojená navíc ještě se správným ohodnocením

proměnné reálného účtu příslušné hladině předchůdce uzlu. Bijekce množiny všech vrcholů grafu na množinu logických proměnných je označována jako BI. Proměnná uzlu implikuje konjunkci proměnné předchůdce tohoto uzlu a proměnné reálného účtu hladiny předchůdce daného uzlu. $RIGHT\_OFFSPRING(v)$ je $TRUE$, právě tehdy, když je vrchol v pravým potomkem. Tyto všechny implikace, logicky kromě implikace kořene stromu, se převedou na CNF a vzniklé klauzule jsou použity v řešiči. Navíc je nutné zahrnout disjunkci proměnných listů, v nichž je hodnota funkce $TRUE$. $REAL\_ACCOUNT\_VAR$ tak představuje zobrazení množiny uzlů grafu G na množinu proměnných reálných účtů, kdy vrcholy na společné hladině zobrazují na stejnou proměnnou. Bohužel se v praxi ukázalo, že pro větší instance je toto řešení nepoužitelné pro exponenciální nárůst uzlů vzhledem k velikosti množiny proměnných reálných účtů. Proto je zapotřebí vytvořený graf komprimovat.

Axiomy principu ohodnocení pomocí formule v CNF

(2.2.45)

$$BI : \{\forall v \in V(G)\} \leftrightarrow VERTEX\_VARIABLES$$

$$REAL\_ACC\_VAR : V \rightarrow ACCOUNT\_VARS$$

$$\forall v \in V(G), v \neq ROOT(G), RIGHT\_OFFSPRING(v) = TRUE :$$

$$BI(v) \Rightarrow BI(PARENT(v)) \wedge REAL\_ACC\_VAR(PARENT(v))$$

$$\forall v \in V(G), v \neq ROOT(G), RIGHT\_OFFSPRING(v) = FALSE$$

$$BI(v) \Rightarrow BI(PARENT(v)) \wedge \neg REAL\_ACC\_VAR(PARENT(v))$$

Vytvoření relace mezi vrcholy

Při vytváření relací mezi předchůdcem a jeho potomkem jde především o vytváření implikací popsaných v předchozí části. Nezávislá proměnná implikace je rovna proměnné potomka a závislá klauzule je tvořena konjunkcí proměnné předchůdce a proměnné reálného účtu, příslušné hladině předchůdce. Pokud je potomek levým potomkem svého předchůdce, je tato proměnná reálného účtu doplněna ještě o svou negaci.

Předchozí postup se uplatní pouze tehdy, pokud má potomek pouze jednoho rodiče, což je typicky na okrajích každé hladiny komprimovaného stromu. V centrálních částech hladin má však jeden potomek dva předchůdce a proto bude postup vytvoření implikací komplexnější. Vrcholy v každé hladině lze totálně uspořádat podle umístění ve frontě této

hladiny. Levý předchůdce daného potomka je vrchol, který byl do fronty předchozí hladiny umístěn bezprostředně před umístěním pravého předchůdce stejného potomka do též fronty. Vzniknou tak dvě relace: levý předchůdce – potomek a pravý předchůdce – potomek. Z obou relací se vytvoří implikace podobným způsobem, jak je uvedeno výše. Rozdíl je v tom, že se v obou těchto implikacích nepoužije na místě nezávislé proměnné proměnná daného potomka, ale jiná pomocná logická proměnná. Poté se vytvoří implikace, jejíž nezávislá proměnná bude proměnná potomka, závislá klauzule bude tvořena disjunkcí nezávislých proměnných (pomocné proměnné) implikací vytvořených pro obě relace.

Komprese stromu

Graf nekomprimované verze obsahuje na jednotlivých hladinách uzly příslušné situacím, kdy je ohodnocen vždy určitý počet proměnných reálných účtů definovaných pro tuto hladinu. Tyto situace vzájemně rozlišují množiny definovaných proměnných podle jejich hodnot, což není nutné. Jedné a téže hladině jsou příslušné situace se stejným počtem kladně definovaných proměnných, ale v jiných proměnných. Důležité je zachovat na každé hladině situace rozlišitelné podle počtu kladně definovaných proměnných. Každá hladina bude obsahovat právě o jeden uzel více než hladina předchozí. Každá hladina bude obsahovat tolik uzlů, kolik je možných různých počtů kladně definovaných proměnných reálných účtů. Celkový počet uzlů bude růst kvadraticky s počtem proměnných reálných účtů, což je již v praxi použitelné i pro velké instance problému.

Algoritmus generování stromu pracuje s množinou proměnných reálných účtů $ACCOUNT\_VARS$ a minimálním počtem účtů k, které musí být pokryty. Běh algoritmu je následující:

1. Vytvoření uzlu, který bude reprezentovat kořen grafu a zařadí jej do nulté hladiny.
2. Přidá do aktuální hladiny hladinu 0.
3. Opakuje kroky 4 až 10 pro hladiny 0 až $|ACCOUNT\_VARS| - 1$.
4. Vytvoří novou prázdnou hladinu.
5. Vezme první vrchol aktuální hladiny, který není listem a zařadí jej do vrcholu aktuálního předka.
6. Pokud je fronta nové hladiny prázdná, vytvoří nový vrchol v nové hladině, který bude reprezentovat situaci po přiřazení hodnoty $FALSE$ do proměnné reálného

účtu příslušné aktuální hladině. Zařadí vrchol na konec fronty nové hladiny a zařadí jej do vrcholu aktuálního potomka.

7. Pokud není prázdná fronta nové hladiny, vezme poslední vrchol z této fronty a zařadí jej do vrcholu aktuálního potomka.
8. Pokud by vrchol aktuálního potomka reprezentoval situaci, kdyby přiřazení hodnoty $TRUE$ do zbylých proměnných nevedlo k vyhodnocení funkce na $TRUE$, označí tento vrchol jako list s hodnotou funkce = $FALSE$.
9. Vytvoří dříve popsané implikace mezi proměnnými vrcholů aktuálního předka a aktuálního potomka.
10. Vytvoří nový vrchol v nové hladině, který bude reprezentovat situaci po přiřazení hodnoty $TRUE$ do proměnné reálného účtu příslušné aktuální hladině. Zařadí uzel na konec fronty této hladiny a zařadí jej do uzlu aktuálního potomka.
11. Pokud by uzel aktuálního potomka reprezentoval situaci, kdy by přiřazení hodnoty $FALSE$ do zbylých proměnných nevedlo k vyhodnocení funkce na $FALSE$, označí tento uzel jako list s hodnotou funkce = $FALSE$ a zařaí její proměnnou do povinné disjunkce, která musí být splněna.
12. Vytvoří dříve popsané implikace mezi proměnnými vrcholů aktuálního předka a aktuálního potomka.
13. Označí novou hladinu za aktuální a přejde ke kroku 3.
14. Vybere z aktuální (= poslední) hladiny proměnné všech uzlů označených hodnotou $TRUE$ a dá je do povinné disjunkce.
15. Konec.

Celkové schéma maximalizačního algoritmu

$act\_covered$ prezentuje počet účtů, které jsou pokryty aktuálně vygenerovanými rolemi. Běh algoritmu je následující:

1. $act\_covered \leftarrow 0$
2. Vytvoří a dá do řešiče formuli, která vygeneruje role pokrývající minimálně $act\_covered + 1$ účtů.
3. Pokud neexistuje nebo není do časového limitu nalezeno řešení, přejde na Konec.
4. Umísti do $act\_covered$ počet pokrytých účtů.
5. Pokud je $act\_covered$ menší než počet všech reálných účtů, přejde na krok 2.
6. Konec.

2.3.3.5 Minimalizační algoritmus

Úlohou minimalizačního algoritmu je nalézt takovou nejmenší množinu rolí, která bude schopna pokrýt jisté minimální procento reálných účtů. Implementovaný algoritmus používá jako podprogram maximalizační algoritmus uvedený v předchozí kapitole.

Schéma minimalizačního algoritmu

Pro nalezení nejmenší množiny rolí je použit $mustCover$ tedy počet účtů, které musí vygenerované role pokrýt, kdy $rolesUsed$ prezentuje aktuální počet použitých rolí s použitím maximalizačního algoritmu $MaxAlg(roleUsed)$. Je stanovena mez $lowerBound$ (dolní) a $upperBound$ (horní) a střed intervalu mezí $middle$. Běh algoritmu je následující:

1. $rolesUsed \leftarrow 1$
2. Pokud $MaxAlg(rolesUsed) = FALSE$, pak $rolesUsed \leftarrow rolesUsed\ x\ 2$. Přejde na 1. krok.
3. $lowerBound \leftarrow 1, upperBound \leftarrow rolesUsed, middle \leftarrow round((lowerBound + upperBound)/2)$
4. Zavolá: $MaxAlg(middle)$
5. Pokud $MaxAlg(middleBound) = TRUE$, $upperBound \leftarrow middle - 1$
6. Pokud $MaxAlg(middleBound) = FALSE$, $lowerBound \leftarrow middle + 1$
7. $middle \leftarrow round((lowerBound + upperBound)/2)$
8. Pokud $lowerBound \leq upperBound$ přejde na krok 4
9. Konec.

2.4 Testování

2.4.1 Testovací podmínky

Testy probíhaly na počítači s procesorem Intel Core i7, 3Ghz s 16 GB operační paměti. Oracle JRE 8 mělo k dispozici 4,6 GB operační paměti.

Role se modelovaly vždy nad čtyřmi atributy, z nichž byl jeden typu multi-value. Bylo ověřeno v praxi, že toto je reálná zátěž programu. Testy byly prováděny nad agregovanými účty, což znamená, že reálné množiny účtů by mohly mít i dvojnásobnou velikost.

Pomocný programový kód, v podobě aplikace nazvanou Role Modeler[28], byl vytvořen v jazyce JAVA. Jazyk Java byl použit z důvodů své multiplatformnosti, rychlosti vývoje v něm vytvářených děl, dostupnosti vývojových nástrojů pro všechny platformy a bohatosti existujících knihoven. První JLDAP[29] knihovna je použita z důvodů open licence. Jedná se o odladěnou a stabilní komponentu umožňující používat technologie LDAP a LDIF. Knihovna také používá vnořené OpenLDAP knihovny jazyka C. Druhá JUNIT[30] je nejpoužívanější testovací knihovnou pro JAVA a byla použita zejména na regresní testy datových struktur. Dále byla zvolena knihovna SAT4j pro implementaci algoritmu MINISAT, což je algoritmus řešící problém SAT pro omezeně velké instance. Knihovně SAT4j se předává celá formule problému, což při větších instancích vede k velké paměťové náročnosti.

2.4.2 Testovací data

Testovací data jsou rozdělena do třech základních kategorií.

1 Vývojová data pro ladění algoritmů.

2 Data pro výkonnostní testy algoritmů.

3 Reálná data z produktivního prostředí.

2.4.2.1 Vývojová data pro ladění algoritmů

První sadu tvoří modelový LDIF soubor s údaji 50-ti zaměstnanců společnosti. Struktura záznamů odpovídá redukované sadě atributů UNIXového systému s hodnotou multivalue v atributu *Groups*.

dn: cn=Public
HomeDirectory: /home/Marwin
Shell: 1
Name: Moro
Surname: Marwin
Groups:135, 142
OrgUnitID: 448
Telephone: 733

Tabulka 1 Ukázkový fragment LDIF souboru první sady (zdroj vlastní)

[28] Podrobnější informace viz Příloha 1 – Nástroj pro modelování rolí
[29] http://www.openldap.org/jldap/
[30] http://www.junit.org/

Druhou sadu tvoří modelový LDIF soubor s údaji 38-ti zaměstnanců. Struktura záznamů odpovídá redukované sadě atributů systému IBM Tivoli Access Manager, který se svoji adresářovou strukturou komunikuje pomocí LDAP. Opět je kladen důraz na existenci multivalue hodnoty v atributu *ertam4groupmember*.

dn: cn=Public
ername : Alena.4515301503137179419
ertam4groupmember: INN_1, NEM_1, INN_L_661, NEM_L_661, INN_3, CSSZ_EMPLOYEE, DKA_A, DKE_A ,NOD_L_0, NOD_USERALL, NEM_14, WRC_1, NEM_4

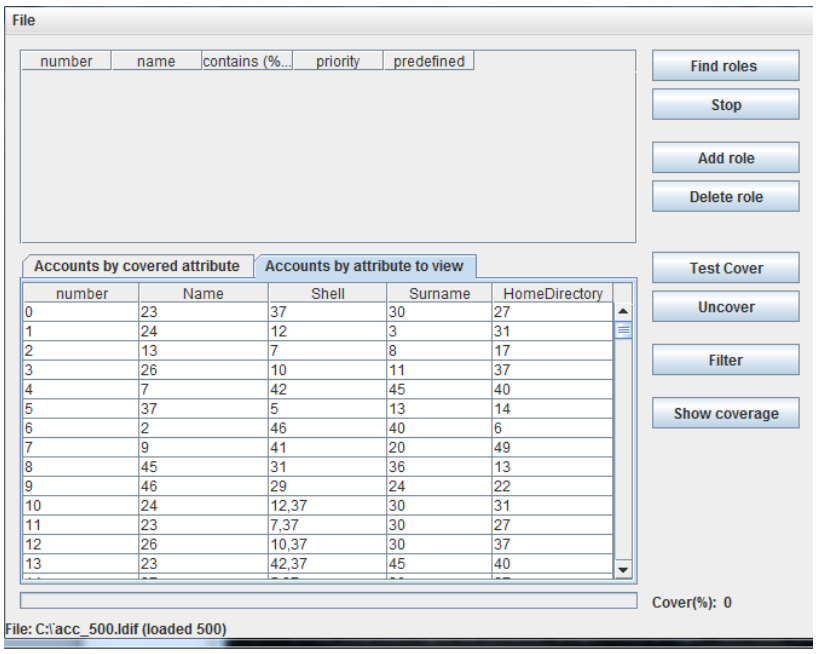
Tabulka 2 Ukázkový fragment LDIF souboru druhé sady (zdroj vlastní)

2.4.2.2 Data pro výkonnostní testy algoritmů

Pro účely výkonnostních testů je zvolena redukovaná sada atributů systému UNIXového typu. Sada byla vygenerována pro 500, 1000 a 2000 účtů virtuálního systému/aplikace.

HomeDirectory: 87
Shell: 69
Shell: 15
Shell: 62
Name: 148
Surname: 178

Tabulka 3 Ukázkový fragment LDIF souboru pro 1000 záznamů (zdroj vlastní)



number	Name	Shell	Surname	HomeDirectory
0	23	37	30	27
1	24	12	3	31
2	13	7	8	17
3	26	10	11	37
4	7	42	45	40
5	37	5	13	14
6	2	46	40	6
7	9	41	20	49
8	45	31	36	13
9	46	29	24	22
10	24	12,37	30	31
11	23	7,37	30	27
12	26	10,37	30	37
13	23	42,37	45	40

Obrázek 13 Načtení výkonnostních testovacích dat – 500 záznamů (zdroj vlastní)

2.4.2.3 Reálná data z produktivního prostředí

V poslední kategorii byla testována i data z produktivních prostředí. Důvodem bylo nejenom dopracování se k reálným katalogům rolí, ale i zajištění stability řešiče při zpracování reálných dat.

ername=Monika.4543917997303823157
ertam4groupmember=eds_1
ertam4groupmember=eds_2
ertam4groupmember=ITIM_GUI_ACCESS
ertam4groupmember= EMPLOYEE
ertam4groupmember=VYP_01
ertam4groupmember=NP2_REF

Tabulka 4 Ukázkový fragment LDIF první sady Access Manager (zdroj vlastní)

eradcontainer=ou=_uzivatele,ou=p1a,ou=pxa
ername=Ludmila.2991532847794093465
eradhomedir=\\si1a6\Home$\1akullud
eradhomedirdrive=H:
eradloginscript=p1a.bat
eradprimarygroup=Domain Users
ergroup=p1aDIS_kukatko_prohlizeni
ergroup=pxa_internet
ergroup=p1aAll
ergroup=Domain Users
erprofile=\\si1a6\Profiles$\1akullud\%profile%

Tabulka 5 Ukázkový fragment LDIF první sady Active Directory (zdroj vlastní)

2.4.3 Nastavené hodnoty expertních nastavení

2.4.3.1 Hill Climb

Hill Climb má ve všech případech nastavených 5 horolezců se zapnutou heuristikou.

2.4.3.2 SAT-based

Přednastavené hodnoty intervalu interního řešiče se řídí následujícími dvěma tabulkami.

Počet účtů	**Maximální interval vnitřního řešiče**
< 500	< 300s
500 – 1000	300 – 500s
> 1000	> 500s

Tabulka 6 Přednastavené hodnoty časového intervalu vnitřního řešiče pro max. problém (zdroj vlastní)

Počet účtů	Maximální interval vnitřního řešiče
< 500	< 60s
500 – 1000	60 – 100s
> 1000	> 100s

Tabulka 7 Přednastavené hodnoty časového intervalu vnitřního řešiče pro min. problém (zdroj vlastní)

2.4.3.3 Genetický algoritmus:

Přednastavené hodnoty expertních nastavení jsou dle následující tabulky.

Parametr	Hodnota
Maximum population size	200
Mutation size	170
Merge size	0
Maximum count of bad instances	5
Hill Climb size	0
Hill Climb ticks	4
Maximum count of the same generation	50

Tabulka 8 Přednastavené hodnoty genetického algoritmu (zdroj vlastní)

2.4.4 Testování maximalizační úlohy

Testovalo se na 500, 1000 a 2000 agregovaných účtech. Pro každou množinu účtů se provedly tři různé testy:

1. Zadalo se takový počet rolí, pro který existuje 100% pokrytí.
2. Zadalo se 9/10 z počtu rolí z bodu 1.
3. Zadalo se 7/10 z počtu rolí z bodu 1.

Pro srovnání jsou udávány i odhadnuté hodnoty délky výpočtu pomocného algoritmu Brutal Force[31].

[31] Brutal Force je pracovní název pro jednoduchý algoritmus, který hrubě vyzkouší všechny možnosti problému. Implementován je z důvodu testování a pro vytvoření přehledu o výkonnosti ostatních algoritmů.

2.4.4.1 500 účtů (Hill Climb, SAT-based, Genetika)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	26s	500
SAT-based	57s	500
Genetika	137s	499
Brutal Force	2.4 mld. let	500

Tabulka 9 Maximalizační úloha na 500 účtech, test č. 1 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	122s	323
SAT-based	412s	315
Genetika	95s	323
Brutal Force	2.3 mld. let	nejvíce možných

Tabulka 10 Maximalizační úloha na 500 účtech, test č. 2 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	61s	113
SAT-based	361s	108
Genetika	53s	97
Brutal Force	2 mld. let	nejvíce možných

Tabulka 11 Maximalizační úloha na 500 účtech, test č. 3 (zdroj vlastní)

2.4.4.2 1000 účtů (Hill Climb, SAT-based, Genetika)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	253s	1000
SAT-based	330s	1000
Genetika	469s	999
Brutal Force	> 6.4 mld. let	1000

Tabulka 12 Maximalizační úloha na 1000 účtech, test č. 1 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	698s	640
SAT-based	1733s	639
Genetika	313s	639
Brutal Force	6.3 mld. let	nejvíce možných

Tabulka 13 Maximalizační úloha na 1000 účtech, test č. 2 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	441s	379
SAT-based	1150s	259
Genetika	210s	373
Brutal Force	5.5 mld. let	nejvíce možných

Tabulka 14 Maximalizační úloha na 1000 účtech, test č. 3 (zdroj vlastní)

2.4.4.3 2000 účtů (Hill Climb, SAT-based, Genetika)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	1403s	2000
SAT-based	1498s	2000
Genetika	1828s	1729
Brutal Force	> 6.4 mld. let	2000

Tabulka 15 Maximalizační úloha na 2000 účtech, test č. 1 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	4265s	1484
SAT-based	4247s	1115
Genetika	1267s	1465
Brutal Force	> 6.4 mld. let	nejvíce možných

Tabulka 16 Maximalizační úloha na 2000 účtech, test č. 2 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet pokrytých účtů
Hill Climb	2038s	699
SAT-based	3210s	633
Genetika	776s	666
Brutal Force	> 6.4 mld. let	nejvíce možných

Tabulka 17 Maximalizační úloha na 2000 účtech, test č. 3 (zdroj vlastní)

2.4.5 Testování minimalizační úlohy

Testovalo se opět na 500, 1000 a 2000 agregovaných účtech. Pro každou množinu účtů se provedly dva různé testy:

1 Zadaly se hledat takové role, aby jich bylo co nejméně a pokrývaly 95% účtů.

2 Zadaly se hledat takové role, aby jich bylo co nejméně a pokrývaly 80% účtů.

Brutal Force je při řešení minimalizačního problému ještě řádově pomalejší oproti řešení maximalizačního problému a je obtížné získat odhad jeho doby výpočtu, z tohoto důvodu je zde opomenut.

2.4.5.1 500 účtů (Hill Climb, SAT-based, Genetika)

Typ algoritmu	Doba výpočtu	Počet nalezených rolí
Hill Climb	40s	10
SAT-based	561s	10
Genetika	120s	10

Tabulka 18 Minimalizační úloha na 500 účtech, test č. 1 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet nalezených rolí
Hill Climb	30s	10
SAT-based	567s	10
Genetika	243s	11

Tabulka 19 Minimalizační úloha na 500 účtech, test č. 2 (zdroj vlastní)

2.4.5.2 1000 účtů (Hill Climb, SAT-based, Genetika)

Typ algoritmu	Doba výpočtu	počet nalezených rolí
Hill Climb	181s	14
SAT-based	1615s	16
Genetika	514s	20

Tabulka 20 Minimalizační úloha na 1000 účtech, test č. 1 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet nalezených rolí
Hill Climb	184s	15
SAT-based	1499s	15
Genetika	1110s	22

Tabulka 21 Minimalizační úloha na 1000 účtech, test č. 2 (zdroj vlastní)

2.4.5.3 2000 účtů (Hill Climb, SAT-based, Genetika)

Typ algoritmu	Doba výpočtu	Počet nalezených rolí
Hill Climb	881s	19
SAT-based	5930s	21
Genetika	2150s	28

Tabulka 22 Minimalizační úloha na 2000 účtech, test č. 1 (zdroj vlastní)

Typ algoritmu	Doba výpočtu	Počet nalezených rolí
Hill Climb	956s	20
SAT-based	5415s	21
Genetika	2636s	38

Tabulka 23 Minimalizační úloha na 2000 účtech, test č. 2 (zdroj vlastní)

2.5 Hodnocení použitých algoritmů

Problémy řešené v tomto výzkumném projektu jsou značně netriviální, a proto bylo navrženo a posléze implementováno více řešitelských algoritmů. Algoritmy jako Hill Climb nebo genetický algoritmus se ukázali i přes svou „nevyzpytatelnost" jako velmi účinné pro hledání dobrých řešení i u takto složitých problémů (Simkova a Tomaskova, 2013, Stepanek a Simkova, 2012).

2.5.1 Hill Climb

Tento algoritmus podává v testech nejlepší výsledky, a to většinou z hlediska času i kvality nalezených řešení. Jediným problémem je rychlé prodloužení doby výpočtu algoritmu Hill Climb, pokud se manipuluje s velkým počtem rolí.

2.5.2 Genetický Algoritmus

Genetický algoritmus podává v testech nejhorší výsledky, co se týče kvality nalezeného řešení, na druhou stranu je poměrně rychlý. Slabším místem u genetického algoritmu je minimalizační problém.

Jeden z problémů, který se v rámci výzkumu řešil je pomalá ohodnocující funkce, kdy její obsluha spotřebuje kolem 99 % času výpočtu. Z toho asi 10 % je kopírování datových struktur ohodnocující funkce pro její urychlení do nově generovaných řešení.

Při implementace genetického algoritmu byl dodržován klasický model populace řešení měnící se po generacích. Standardní metody generování nových řešení jsou mutace a křížení, přičemž existuje mnoho názorů o vhodnosti použití jednoho nebo druhého. Testy ukázaly, že křížení je nevhodné a jen zpomaluje algoritmus. Na druhou stranu je možné vyzkoušet při křížení pohlížet na role jako celky, místo křížení na úrovni atributů. Sice dochází k eliminaci některých možností, zanechává tedy méně možností, ale když se vezme v úvahu, že na prozkoumání všech možností stačí mutace, mohla by granularita na úrovni rolí být užitečná.

Hledání řešení při minimalizačním problému by výrazně urychlila možnost urychleného zjišťování pokrytí nejen pro změnu jedné role, ale i pro odebrání nebo přidání jedné role. Maximalizační problém, kde je počet rolí fixní, je z tohoto hlediska více vhodný.

Testování ukázalo některé vlastnosti těchto nastavení:

- Častější používání metody spojení vede k výrazně méně přesným výsledům
- Je vhodné, aby při selekci bylo smazáno mnohem více řešení, než jich má být ponecháno. Poměr ponechaných a smazaných řešení, (dá se říci poměr maximální velikost populace a velikosti mutace) určuje, kolik bude průměrně z jednoho jedince vygenerováno nových řešení. Ukázalo se, že by tento poměr měl být určitě větší než jedna, i větší než pět přináší dobré výsledky.
- Počet ponechaných řešení po selekci udává, jak různorodá může být další generace, toto číslo by nemělo být příliš malé, méně než deset není doporučováno.
- Maximální velikost populace by neměla být příliš velká, čím je větší, tím déle trvá jedna generace, a to vede k celkovému prodloužení doby výpočtu. Nad tisíc není doporučováno. Toto by však nemělo mít vliv na kvalitu dosažených výsledků.
- Maximální počet generací bez zlepšení by neměl být příliš malý, aby měla populace čas na objev lepšího řešení k pokračování. Počet nižší než padesát by mohl vést ke zbytečnému utnutí algoritmu, i když by pořád ještě mohl úspěšně pokračovat. Příliš velký počet vede k prodloužení doby výpočtu, ale jen tím, že ho nechá běžet déle. Je možné toto číslo zadat vysoké a přerušit algoritmus ručně. Tím získá uživatel plnou kontrolu nad dobou trvání algoritmu. Buď nechá program běžet dlouhodobě, či ho ukončí až podle momentálního dojmu o kvalitě výsledku.

2.5.3 SAT-based

Vypracovaní specifikace a vlastní implementace algoritmu, který převede problém pokrytí účtů rolemi na booleovskou rovnici, kterou poté poskytne knihovně implementovaného SAT řešiče a vrácené výsledky transformuje zpět do sémantiky vstupního problému, bylo poměrně dosti časově náročné. Přitom hlavní problémy, které se v procesu vývoje vyskytly, se v naprosté většině případů vztahovaly spíše k implementačním obtížím než k teoretickým záležitostem. Konečné zúčtování, které provedly vygenerované testy, působí subjektivně mírně optimistickým dojmem.

Výhody:

- možnost uživatelsky nastavit rychlost vs. přesnost algoritmu,
- poměrně vysoká přesnost i při nižším nastaveném času řešení,
- algoritmus lze lehce upravit pro budoucí použití s rychlejšími SAT řešiči, jejichž vývoj je velmi dynamický.

Nevýhody:

- velká paměťová náročnost,
- velká vlastní režie algoritmu - přímo plyne z první nevýhody,
- existence časových stochastických intervalů; nastavení času hodnotami z tohoto intervalu může ovlivnit výsledky pro stejnou instanci problému. Tyto intervaly jsou specifické pro každou unikátní instanci problému.

3 Provozní informační systémy

Tato krátká kapitola vytváří propojení mezi předešlými kapitolami a navazující kapitolou o návrhových vzorech.

Metody řízení přístupu jsou základním prvkem realizace veškerých IS. V obecném pohledu zajišťují odpovídající přístupy k datům a k exekutivním činnostem IS podle definovaných pravidel. Tuto činnost lze realizovat různými způsoby – v souvislosti s problematikou RBAC je možno konkrétní realizace popsat následujícím způsobem.

Provozní informační systémy, které řeší vnitřní informace podniku, a realizace RBAC lze obecně rozdělit na realizaci RBAC na databázovém serveru, např. (Rashid et al., 2010), na distribuovaných databázích, např. (Almutairi et al., 2012), na business vrstvě, cloudové řešení, např. (Krutz et al., 2010, Younis et al., 2014). Dva nejčastější přístupy jsou popsány dále, a to realizace na business vestě a v rámci databáze.

3.1.1 Model realizace RBAC na business vrstvě

Základními stavebními kameny jsou role a oprávnění. Oprávnění lze definovat jako vazby mezi transformačními procedurami a objekty. V pojetí business vrstvy lze obecně definovat oprávnění jako spárování objektů a jejich metod, případně tříd a jejich metod, případně table queries a view queries – záleží na konkrétní modelaci návrhu business vrstvy. Výsledná oprávnění jsou tak přiřazena rolím, ke kterým jsou následně přiřazeni uživatelé. V souladu s hlavní myšlenkou práce tak lze uvažovat RBAC jako řízení přístupů, které popisují komplexní přístupovou politiku, redukují chyby v administrativní politice a náklady administrativy. V business vrstvě realizovaného IS je zapotřebí definovat základní komponenty RBAC, tedy vazby:

- Role – Permissions.
- Role – Role (zajišťuje rekurzivitu modelu).
- User – Role.

Analytik musí zajistit model, který bude tyto komponenty využívat a zároveň konceptuálně realizovat celé pojetí RBAC.

Na rozdíl od realizace RBAC modelu na úrovni databázového serveru viz dále, business vrstva (BL[32]) tak splňuje podmínku *Data Abstraction*, což přímo vychází z objektového pojetí BL.

Při návrhu konkrétní realizace je třeba respektovat opačné směřování hierarchii rolí a oprávnění. To je postup, který zajišťuje metodu nejnižšího oprávnění, na které je RBAC stavěno.

Z pohledu modelu lze hierarchii rolí rozdělit na dva základní případy, a to obecnou hierarchii a omezenou neboli limitovanou.

Obecná hierarchie má základní vlastnost z pohledu objektového modelu, a to podpora *Multiple Inheritance*. To znamená, že role postupně „nabalují" svoje práva a neexistuje role, která by nenavazovala na některého předka, samozřejmě s výjimkou bázových.

Limitovaná hierarchie je složitější, ale jedná se v praxi podobnější model, který umožňuje soustřeďovat více obecných hierarchií. S dostatečným zobecněním lze prohlásit, že tento právě model omezující hierarchie bude blíže praktické realizaci, jelikož také umožňuje skládání rolí. Pro odstranění konfliktů v této hierarchii je zapotřebí definovat buď statické nebo dynamické oddělení rolí tzv. duty roles.

Základní prvky RBAC v business vrstvě jsou uživatelé, role, operace, objekty. Operacemi jsou přidání anebo odstranění uživatele a přidání anebo odstranění role. Dále vytvoření sezení včetně přidání aktivní role, odstranění aktivní role nebo kontrola přístupu. Součástí jsou dále funkce, tedy metody vracející hodnotu a vazební metody. Za pomocí těchto objektů, metod a funkcí lze pak realizovat RBAC model na straně business vrstvy.

Funkce:

- AssignedUsers(Role) - množina uživatelů přiřazených k roli.
- AssignedRoles(User) – množina rolí přiřazena uživatelům.
- RolePermissions(Role) – množina oprávnění přiřazených roli.
- UserPermissions(User) – množina oprávnění přiřazených uživateli.
- SessionRoles(Session) – množina rolí přiřazených k aktuálnímu sezení.
- SessionPermissions(Session) – množina oprávnění přiřazených aktuálnímu sezení.

[32] Business Layer

- RoleOperationOnObject(Object) – množina operací role vykonávaných na objektu.
- UserOperationOnObject(Object) – množina operací uživatele vykonávaných na objektu.

Vazební metody:

- AddInheritance(Role1, Role2) – přidání dědičnosti.
- DeleteInheritance(Role1, Role2) – odebrání dědičnosti.
- AddAscendant(Role) – vytvoří roli k Role jako potomka.
- AddDescendant(Role) – vytvoří předchůdce dané role.

Administrativní funkce

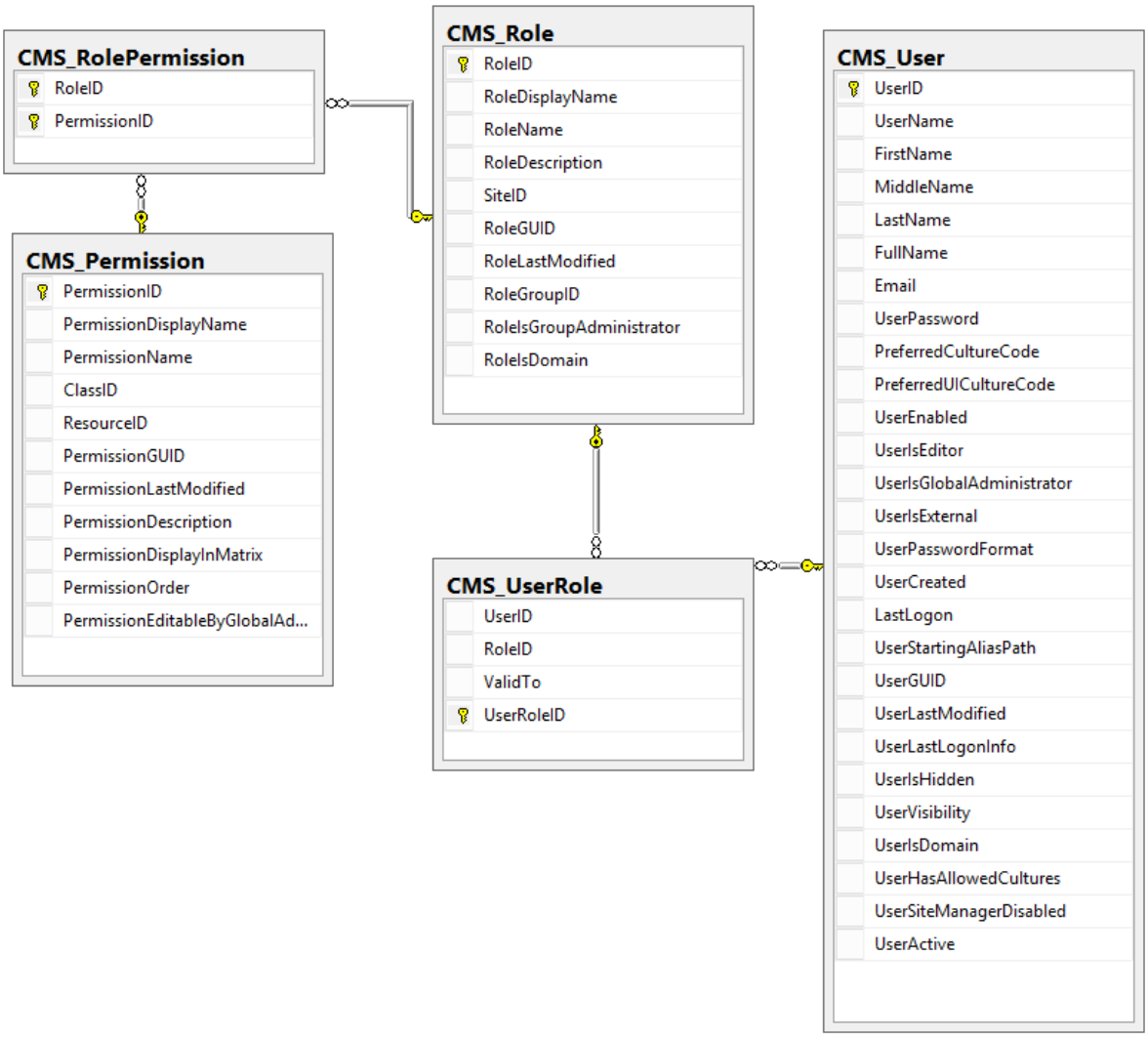
- AuthorizedUsers(Role) – autorizovaní uživatele dané role.
- AuthorizedRoles(User) – autorizované role daného uživatele.
- CreateSSDSet - vytvoří instanci pro SSD.
- DeleteSSDSet – odstraní instanci pro SSD.
- AddSSDRoleMember - přidá roli k SSD.
- DeleteSSDRoleMember – odstraní roli k SSD.
- SetSSDCardinality – nastaví SSD kardinalitu.
- SSDRoleSets – vrátí sady SSD rolí.
- SSDRoleSetRoles – vrátí množinu rolí spojených s pojmenovanou SSD rolí
- SSDRoleSetCardinality – vrátí kardinalitu množiny v rámci pojmenované SSD role, pro kterou platí omezení ve stejné relaci.
- CreateDSDSet - vytvoří instanci pro DSD.
- DeleteDSDSet – odstraní instanci pro DSD.
- AddDSDRoleMember - přidá roli k DSD.
- DeleteDSDRoleMember – odstraní roli k DSD.
- SetDSDCardinality – nastaví DSD kardinalitu.
- DSDRoleSets – vrátí sady DSD rolí.
- DSDRoleSetRoles – vrátí množinu rolí spojených s pojmenovanou DSD rolí
- DSDRoleSetCardinality – vrátí kardinalitu množiny v rámci pojmenované DSD role, pro kterou platí omezení ve stejné relaci.

3.1.2 Model realizace RBAC na databázi

Tato podkapitola popisuje řešení RBAC na databázi resp. databázích, nikoliv ovšem samotné role databázového serveru, cílem stejně jako předešlého řešení je použití RBAC v rámci informačního systému nebo aplikace (Zelenka a Borkovcova, 2016).

Řešení, které nepoužívá třívrstvý model, má většinou tyto technické důvody:

- Důraz na okamžitý výkon aplikace.
- Důraz na množství aktivních uživatelů.
- Důraz na množství přenášených dat.



Obrázek 14 Ukázka realizace RBAC v databázi – redakční systém (zdroj vlastní)

Mezi takové konkrétní příklady patří rezervační systémy (hotelové, vstupenky apod.) anebo technologické informační systémy, měřící a vyhodnocující velké množství dat z technologických vstupů – příkladem mohou být informační systémy firmy Boeing, kde

pro lineární online kontroly motorů vstupují terabyty dat během hodiny. V takových systémech je zapotřebí minimalizovat počet vrstev a využít schopností a výkonu samotného databázového serveru. Vzhledem k zachování zabezpečení je vhodné zachovat i zde principy RBAC. Základní návrh objektů vychází ze základní definice RBAC.

Atributy jednotlivých tabulek lze libovolně modifikovat, dle konkrétních potřeb daného informačního systému. Například uživatel může mít navíc omezený čas práce s informačním systémem, nebo některé role jsou vázány na další podmínky – vycházející například ze stavu dokladu v procesním schématu dané aplikace. Konkrétní realizace pak navazuje na potřebu stejných objektů, tříd, metod a funkcí tak, jak byly popsány v třívrstvé realizaci s tím rozdílem, že tyto entity jsou realizovány přímo v databázovém serveru.

4 Návrhové vzory RBAC

V rámci a v návaznosti na projekt *„Napojení systému pro správu identit na algoritmy pro analýzu přístupových oprávnění a modelování uživatelských rolí“* proběhla analýza uživatelských oprávnění ve spolupracujících organizací, postupně ale i s organizacemi původně vzdálenými. Mezi sledované údaje či procesy patřila organizační struktura; klíčové případy užití a aktéři vstupující do systému; strategické a klíčové procesy a celkové řešení řízení uživatelských přístupů. V rámci testovací fáze (kapitola 2.4) navrženého systému pro správu identit bylo vyhodnoceno několik případových studií. Případové studie byly řešeny v různých kategoriích ekonomických činností.

Všechna získaná data byla šifrována, jednalo se pouze o LDIF soubory, LDAP připojení z důvodu bezpečnosti nebylo možné. LDIF soubor obsahoval převážně účty, skupiny a sledované parametry. Konkrétní identity jednotlivých uživatelů zůstaly skryté, a to z důvodu ochrany osobních údajů a samozřejmě vnitřní politiky bezpečnosti jednotlivých organizací.

Pro lepší pochopení je uveden modelový příklad:

Identita:

- Alena, vedoucí přestupkového oddělení Magistrátu města „Zlatá Louka“, pozice je zahrnuta v organizační struktuře, pracovní zařazení: vedoucí.

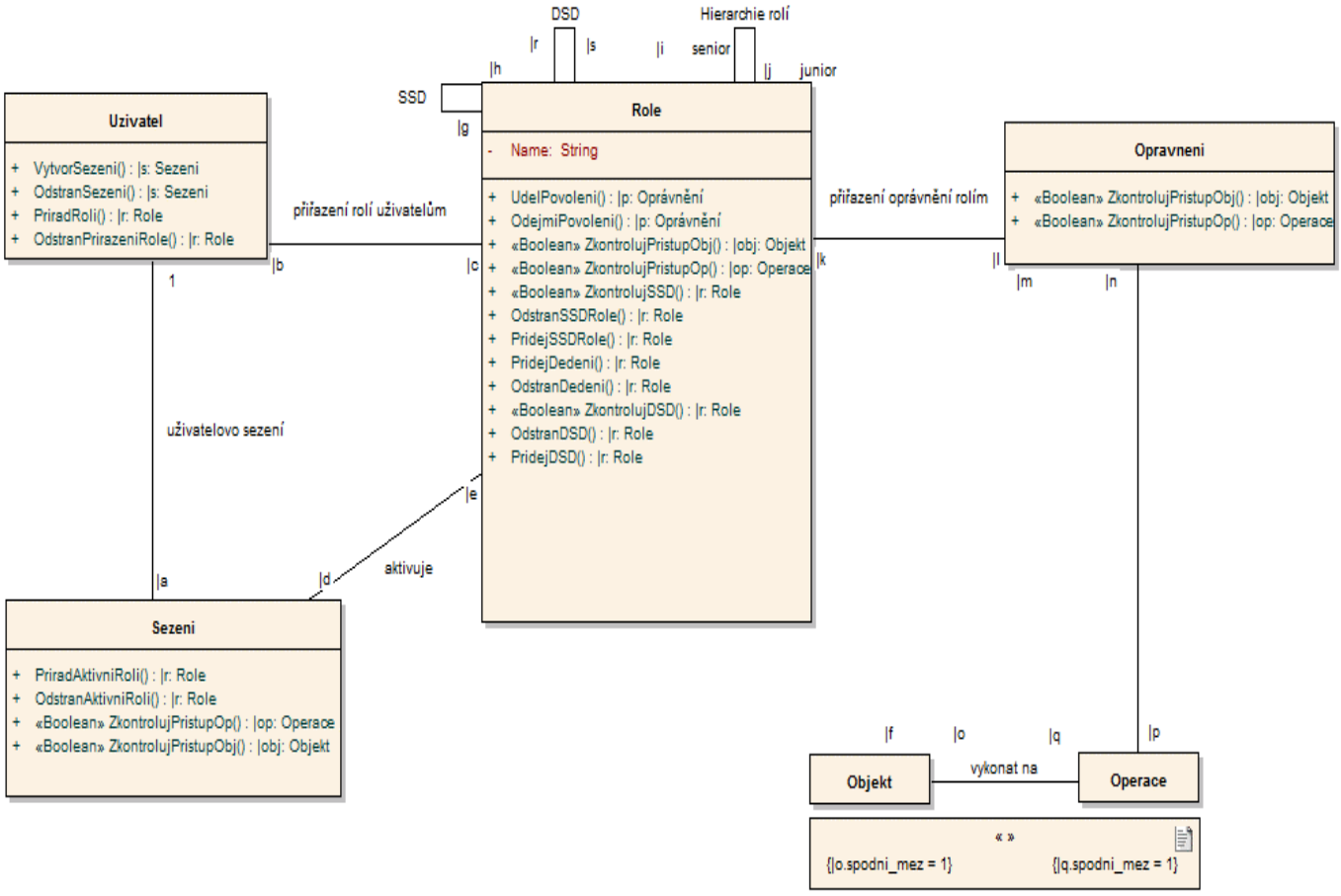
Uživatel- dodaný v LDIF souboru:

- dn: cn=PUBLIC
- ername : Alena.4515301503137179419
- ertam4groupmember: INN_1,NEM_1,INN_L_661,NEM_L_661,INN_3,CSSZ_EMPLOYEE,DKA_A,DKE_A,NOD_L_0,NOD_USERALL,NEM_14,WRC_1,NEM_4

4.1 RBAC návrhové vzory

Na základě současných RBAC vzorů a různých technik k jejich tvorbě, bylo vybráno několik vhodných postupů k provázání na správu identit pro navržení konkrétních modelů dle případových studií. Jednotlivé použité stěžejní části jsou popsány níže. První možný přístup

pro začlenění specifikace RBAC do modelů UML popisuje (Kim et al., 2004).



Obrázek 15 RBAC diagram tříd - šablona, vychází z (Kim et al., 2004) (zpracování vlastní)

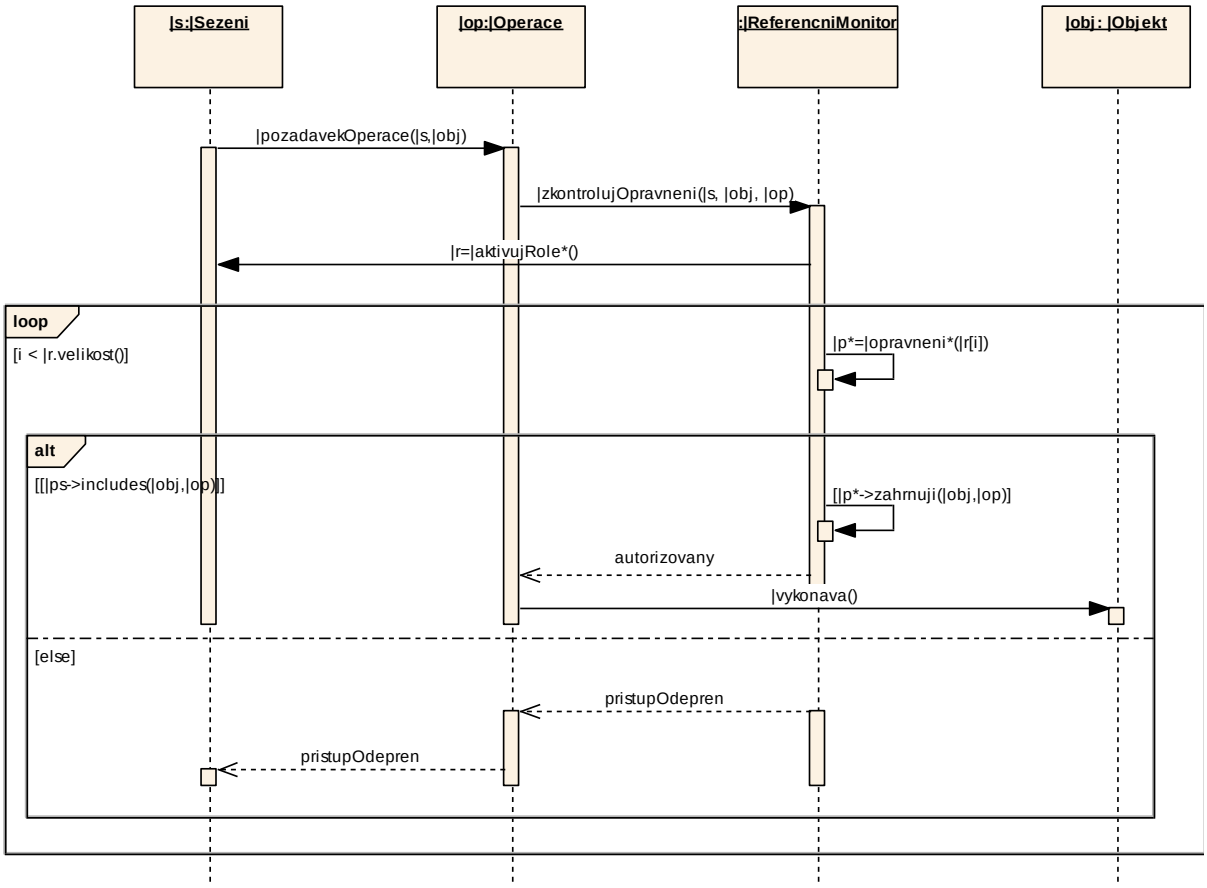
Obrázek 15 popisuje UML šablonu diagramu tříd pro použití RBAC včetně hierarchie rolí, SSD a DSD. Výše uvedená šablona obsahuje parametry (zástupný symbol |), které jsou asociovány s atributy a operacemi. Omezení uvedené pod Objekt a Operace znamená multiplicitu 1, která může být spojena s parametrem o a q (|o, |q).

Aplikování této šablony do RBAC modelu vyžaduje:

1. Tvorbu primárního modelu, který reprezentuje jednotlivé třídy informačního systému organizace bez atributů a operací.
2. Získání obsahově specifického diagramu aplikací instance šablony RBAC a propojení parametrů primárního modelu se šablonou RBAC.
3. Vytvoření kompozitního diagramu představuje sloučení primárního modelu s obsahově specifickým diagramem a získání výsledného modelu aplikovaného na vybraný informační systém.

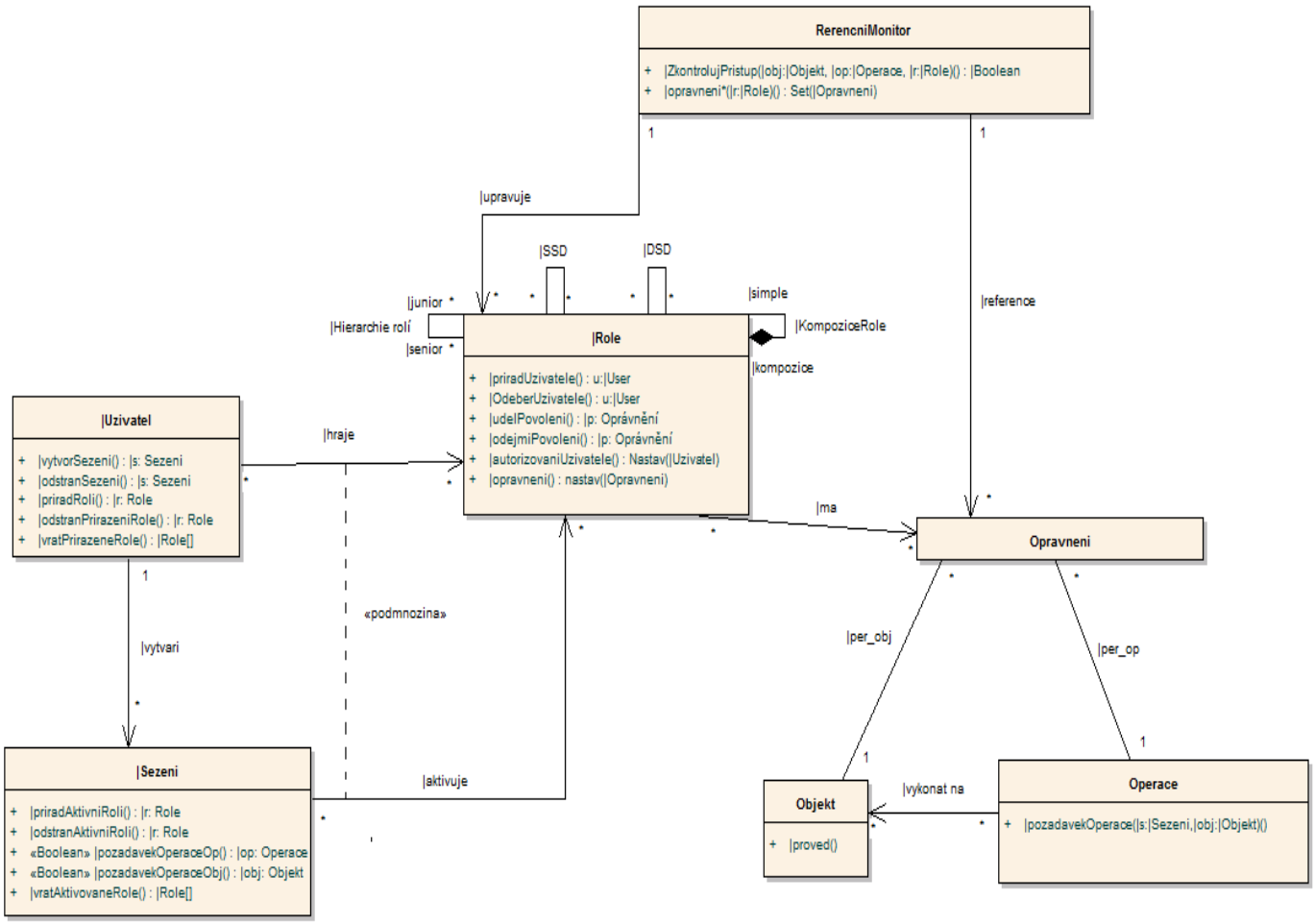
4. Popsání čtyř principů RBAC na konkrétní vzor, a to základního jádra, hierarchie, SSD a DSD.

V návazném článku (Kim et al., 2006) popisuje rozšíření UML nazvané RBML pro reprezentaci struktury a chování vzorů s možností zachycení instancí vzorů, dochází k doplnění sekvenčního diagramu v RBAC respektive RBML. Na obrázku 16 je zachycen objekt, operace, oprávnění, subjekt a referenční monitor. První čtyři základní pojmy jsou rozšířeny o tzv. referenční monitor, který kontroluje omezení SSD, DSD a hierarchii rolí.



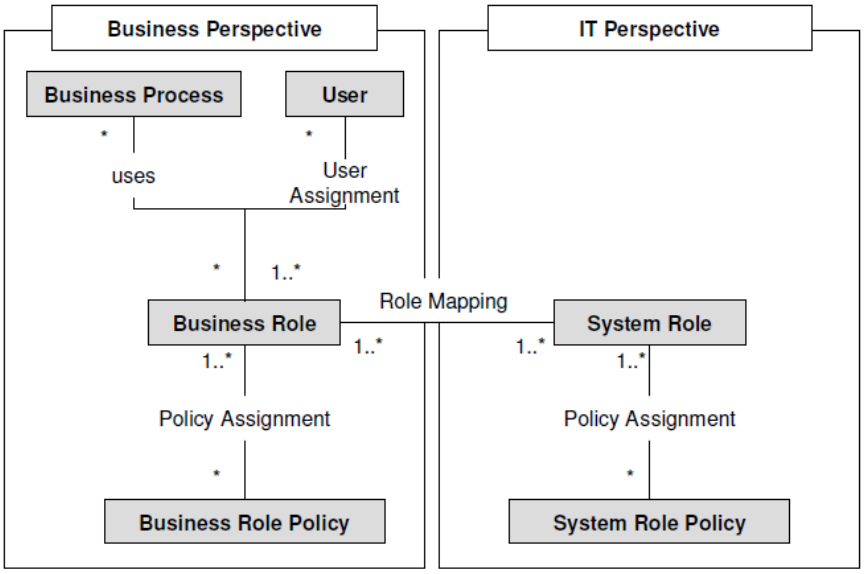
Obrázek 16 RBAC interakce, vychází z (Kim et al., 2006) (zpracování vlastní)

Obrázek 16 popisuje požadavek na operaci vyvolaný v rámci sezení a ten je zachycen referenčním monitorem ke kontrole dostupnosti. Požadavek je tedy kontrolován proti sadě oprávnění každé role v rámci sezení, v případě true je povolen, v případě false je zamítnut.



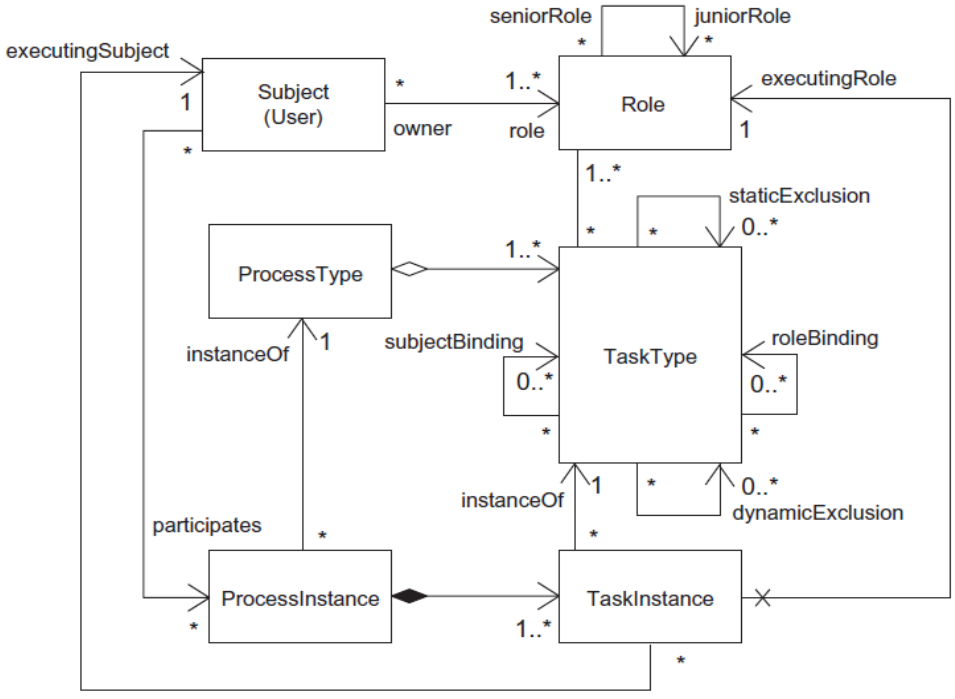
Obrázek 17 RBAC – struktura řešení, vychází z (Kim et al., 2006) (zpracování vlastní)

(Klarl et al. 2009) popisuje business role a modelování podnikových procesů. Uvádí, že běžné podnikové procesy odrážející organizační řád jsou jednotlivé úkoly a pracovní profily, kdy úkolem je činnost, která se provádí pravidelně a pracovní profily jsou organizační klasifikace, seskupování zaměstnanců se stejnými schopnostmi a odpovědnostmi. To vše popisuje standardní obchodní koncept, business role by měly být zásadní ve vztahu k použitým informačních technologiím v rámci organizace. Na následujícím obrázku 18 je uveden B&S-RBAC jako vzor pro propojení obchodních a systémových rolí.



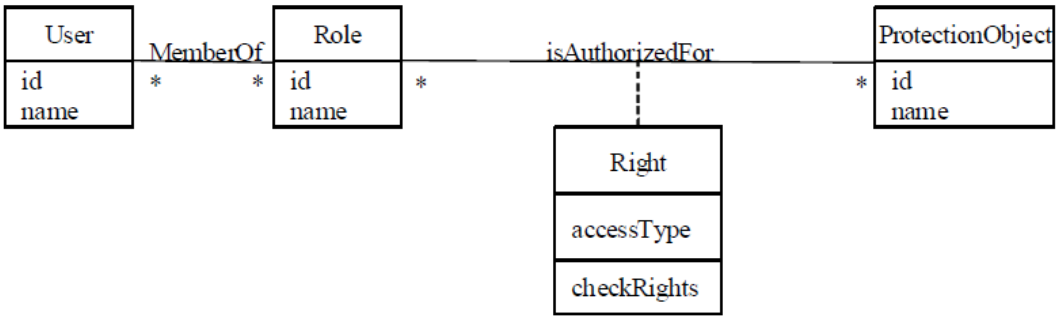
Obrázek 18 B&S-RBAC (Klarl et al., 2009)

Systémové role jsou zde přiřazeny obchodním (business) rolím, nikoliv jednotlivým uživatelům. Zaměření se na konkrétní aktivity v systému zpracoval (Strembeck a Mendling, 2011). Ti pomocí UML diagramu aktivit navrhli koncept s hlavními elementy modelu firemních aktivit RBAC a dále vytvořili rozšíření pro UML diagram aktivit ve spojení s RBAC.

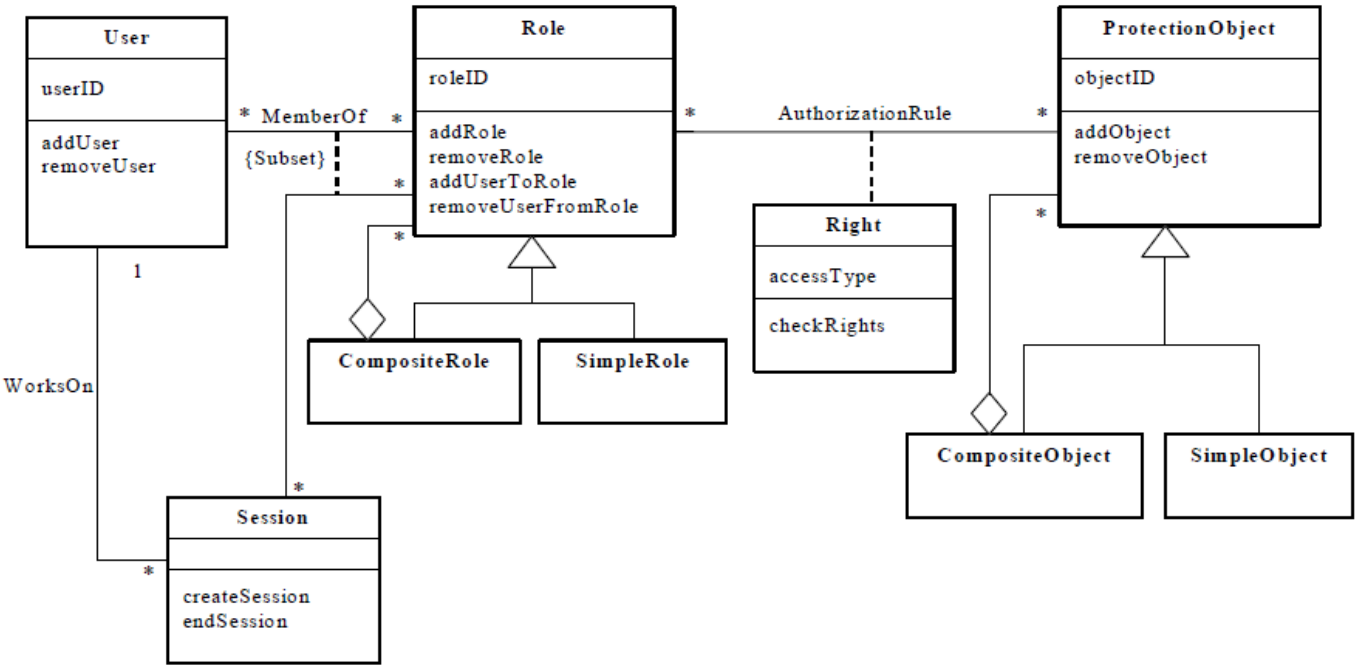


Obrázek 19 Model firemních aktivit RBAC (Strembeck a Mendling, 2011)

Pro doplnění je vhodné uvést i (Fernandez et al., 2008), kteří popsali definičně návrhové vzory pro řízení přístupů včetně RBAC, jak je vidět níže na uvedených obrázcích.



Obrázek 20 Základní RBAC návrhový vzor (Fernandez et al., 2008)



Obrázek 21 Základní RBAC návrhový vzor (Fernandez et al., 2008)

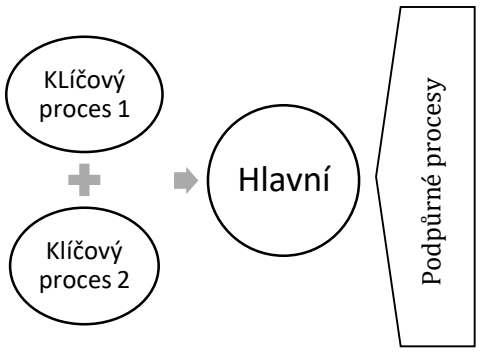
4.2 Analýza procesů a organizační struktury

Jak je již výše uvedeno, business role by měli v ideálním případě odpovídat systémovým rolí, to ovšem v praxi není plně dodržováno. Z tohoto důvodu bylo nutné u každého subjektu identifikovat skutečnou organizační strukturu včetně pracovních rolí v tomto kontextu i pracovních pozic a zároveň zohlednění pracovních skupin. Dále došlo k identifikaci zkoumaných procesů dané organizace či instituce a ty byly popsány na kartě procesů. Pro přehlednost jsou vybrané karty procesů týkající se neziskových organizací, pro autorku nejbližších terciálnímu vzdělávání, v příloze č. 2.

4.2.1 Identifikace činností

Identifikace činností vychází z knih, článků a jiných osobních děl prof. Ing. Řepy Václava, CSc. Pro přístup k identifikaci je použit rámcový procesní model, který využívá k rozdělení procesů strukturu obsahující hlavní proces, klíčový proces a podpůrné procesy. Výhody tohoto modelu jsou již zřejmé z jednoduššího pohledu na procesy a větší přehlednost.

V rámci procesního řízení hraje velkou roli zpracování rámcového procesního modelu, který přehledně a srozumitelně popisuje identifikované procesy probíhající v celkovém kontextu dané organizace. Jednotlivé aktivity zmapovaných procesů jsou definovány v podpůrných procesech identifikovaného hlavního procesu. Pokud organizace neměla identifikované základní procesy, hlavním úkolem bylo identifikovat základní procesy v rámci organizace a později nalézt úroveň vyhovující zpracování procesů pro danou organizaci. Cílem identifikace procesů je přehledná forma. Každý proces je identifikován základními údaji na své kartě.



Obrázek 22 Zjednodušený rámcový procesní model (zdroj vlastní)

Základem celého postupu je identifikace procesu:

- Klíčové výstupy, klíčový vstupy, transformace: nutné podmínky, činnosti, produkty.

Karta procesu obsahuje:

- *Cíl procesu*, který představuje, k čemu nám daný proces slouží a čeho je průběhem tohoto procesu dosahováno.
- *Popis procesu*, který detailně rozvijí proces.
- *Přidaná hodnota*, která představuje formu naplnění daného procesu.
- *Zákazník procesu*, který představuje odběratele daného procesu.
- *Vlastník procesu*, který představuje pracovníka, který je zodpovědný za celý průběh daného procesu, za jeho korektnost a správnost.
- *Spolupráce v rámci organizace* představuje kooperaci při průběhu aktivit procesu.

- *Klíčové vstupy*, které představují veškeré vstupy potřebné ke spuštění daného procesu.
- *Klíčové výstupy*, které představují to, co je produktem daného procesu.
- *Podpůrné procesy* představují sub procesy logicky patřící do hlavního procesu.
- *Posloupnost činností* slouží k jednoznačnému určení kroků naplnění konkrétního procesu.
- *Časové trvání průběhu činnosti a režim aktivit* definuje časovou náročnost a pravidelnost procesu.
- *Podpora ICT* určuje pomocné prostředky využité k naplnění procesu.

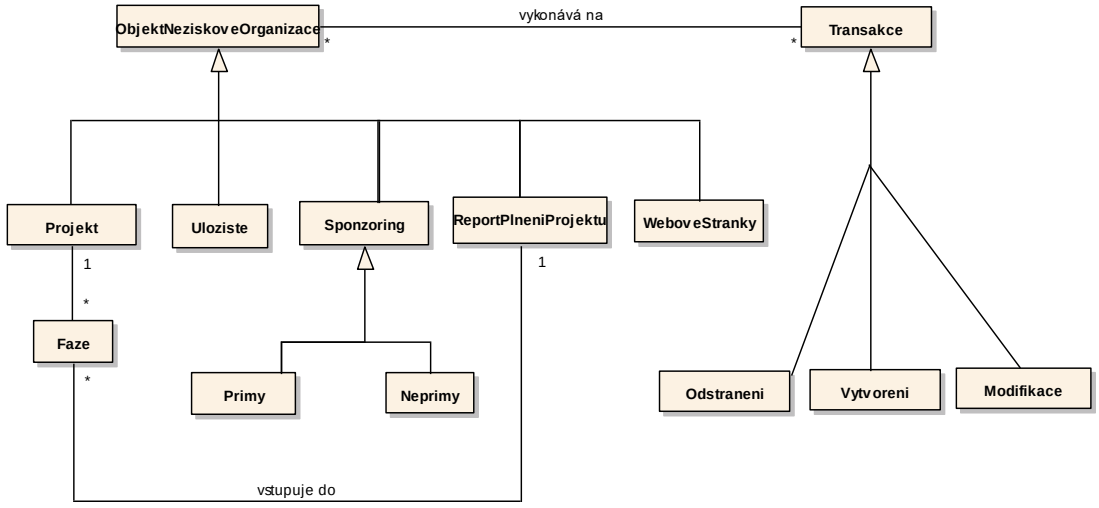
4.3 Aplikace návrhových vzorů RBAC

Níže jsou uvedeny vybrané jednotlivé aplikace návrhových vzorů RBAC na konkrétní vybranou problematiku nejčastěji řešenou v rámci dané ekonomické či technicko-provozní činnosti. Aplikace návrhových vzorů je odvozena od samotné analýzy procesů v rámci organizace a dále od výstupní sady rolí výše uvedeného testování navržené aplikace provázané na vybrané algoritmy. Jednotlivé kroky nejsou již v rámci jednotlivých kategorií podrobně popsány, a to z důvodu, že vycházejí právě z již výše popsaných návrhových vzorů. Ve spojení s výstupní sadou rolí vygenerovanou testovací aplikací byla zvolena kombinace parametrizovaného UML modelu v návaznosti na inovovaný návrhový vzor s referenčním monitorem a byla zakomponována samostatná analýza procesů a identifikace jednotlivých činností. V závěru aplikace došlo k úpravám ve vztahu k business rolím dané organizace.

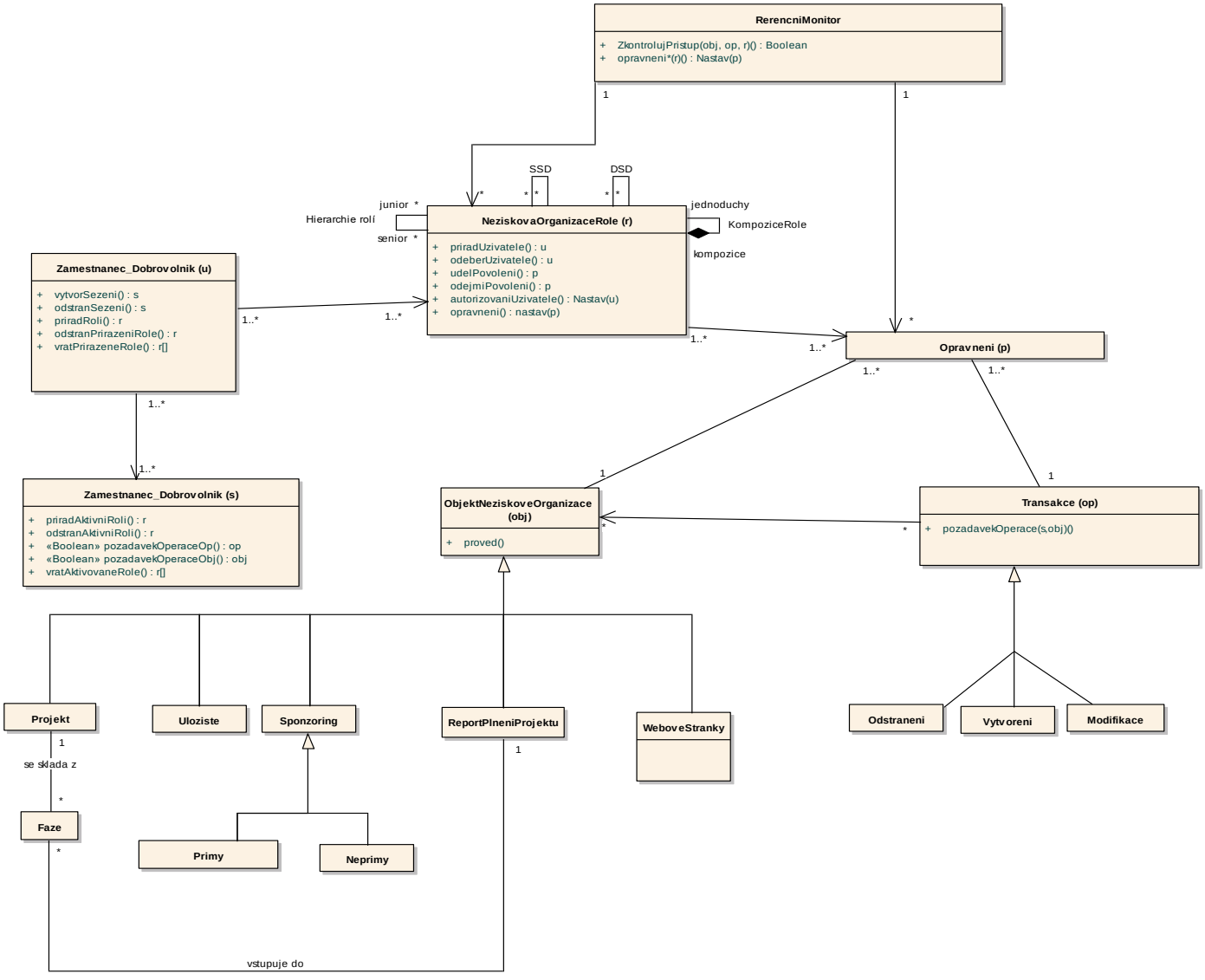
Ukázkové aplikace návrhových vzorů:

- Do prvního okruhu spadají neziskové organizace, u kterých bylo zjištěno, že se jejich klíčové procesy bez ohledu na zaměření velice prolínají a podobají.
- Druhý okruh byl odvozen od klasifikace ekonomických činností CZ-NACE. V rámci něhož bylo zkoumáno několik oblastí. Níže jsou uvedené vybrané, a to z důvodu rozsáhlosti:
    - 55.1 Ubytování v hotelích a podobných ubytovacích zařízeních.
    - 65 Pojištění, zajištění a penzijní financování, kromě povinného sociálního zabezpečení

4.3.1 Neziskové organizace



Obrázek 23 Primární model – nezisková organizace (fáze 1) (zdroj vlastní)

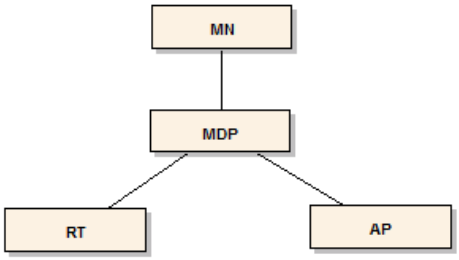


Obrázek 24 Složený model – nezisková organizace (fáze 3 spojená s fází 2) (zdroj vlastní)

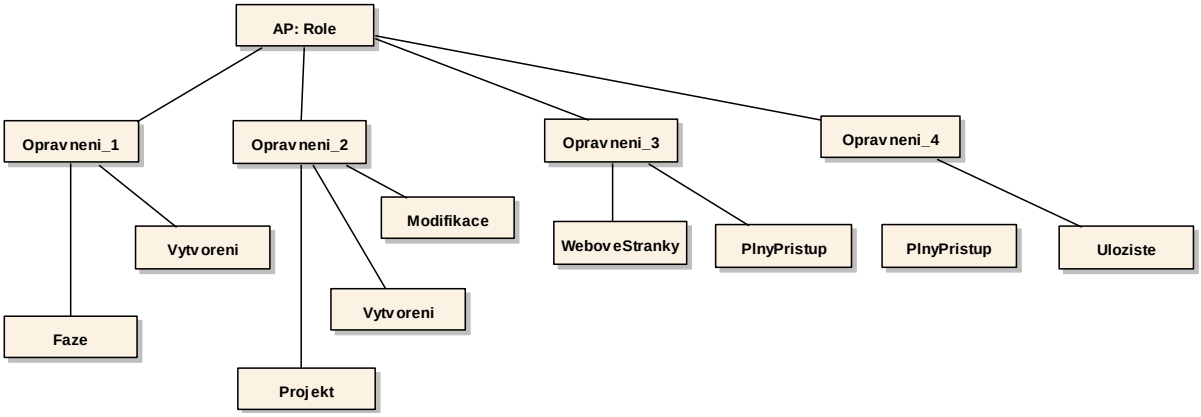
4.3.1.1 Základní jádro

Instance NeziskovaOrganizaceRole:

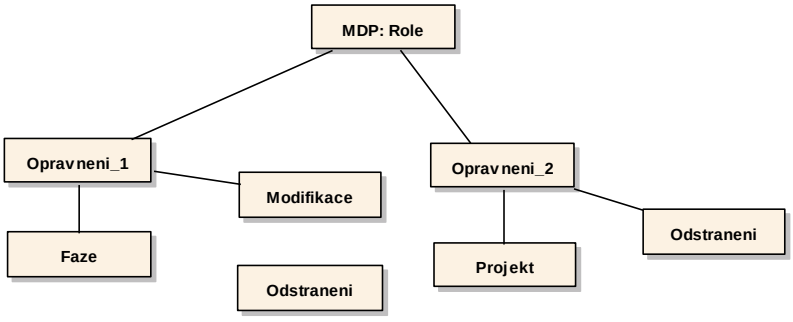
- administrativní pracovník/ce (AP), manažer dotačního projektu (MDP), manažer nadace (MN), realizační tým (skupina RT).



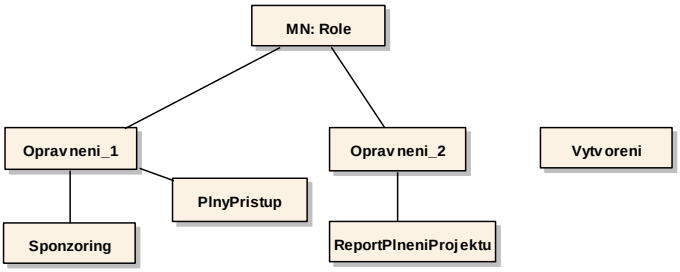
Obrázek 25 Organizační struktura – neziskové organizace (zdroj vlastní)



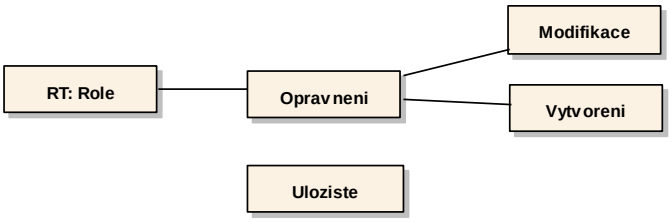
Obrázek 26 Základní jádro RBAC – neziskové organizace (fáze 4 – administrativní pracovník/ce) (zdroj vlastní)



Obrázek 27 Základní jádro RBAC – neziskové organizace (fáze 4 – manažer/ka dotačního proj.) (zdroj vlastní)

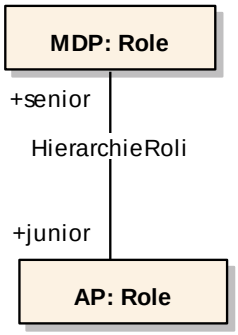


Obrázek 28 Základní jádro RBAC – neziskové organizace (fáze 4 – manažer/ka neziskové org.) (zdroj vlastní)

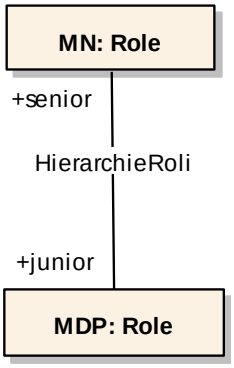


Obrázek 29 Základní jádro RBAC – neziskové organizace (fáze 4 – realizační tým (skupina) (zdroj vlastní)

4.3.1.2 Hierarchie

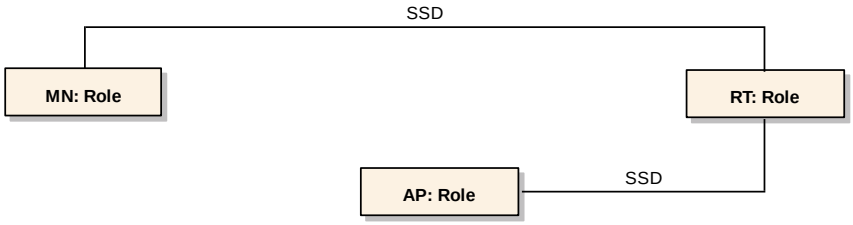


Obrázek 30 Hierarchie RBAC – neziskové organizace (fáze 4 – MDP-AP) (zdroj vlastní)



Obrázek 31 Hierarchie RBAC – neziskové organizace (fáze 4 – MN-MDP) (zdroj vlastní)

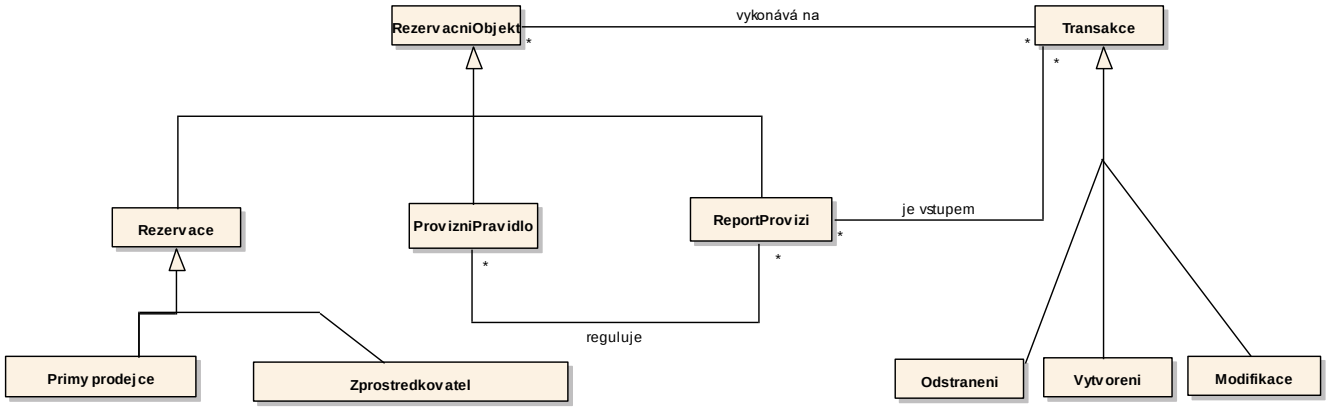
4.3.1.3 SSD, DSD



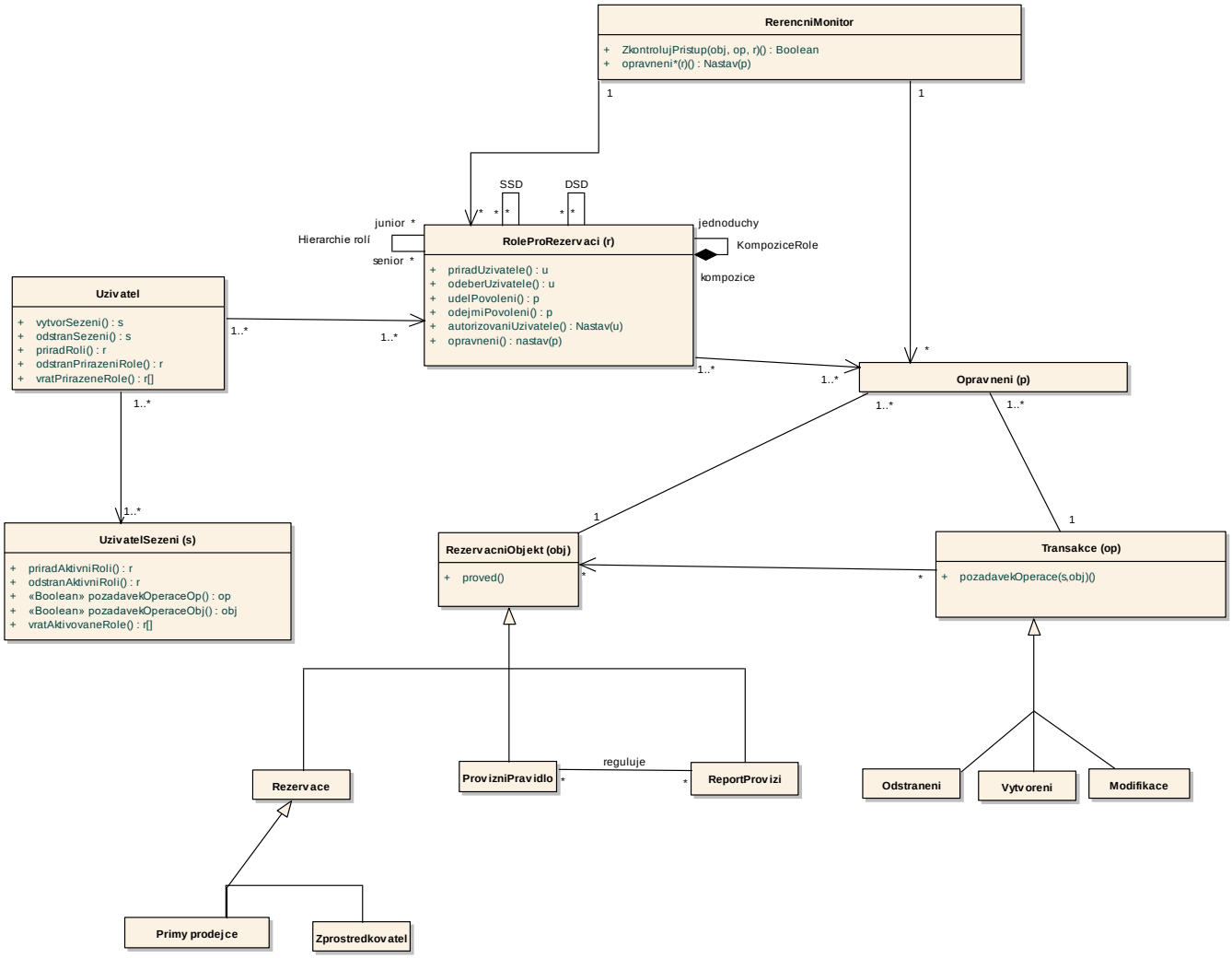
Obrázek 32 SSD, DSD RBAC – rezervace v ubytovacích zařízeních (fáze 4) (zdroj vlastní)

4.3.2 Ubytování v hotelích a podobných ubytovacích zařízeních

Klíčovým procesem v ubytovacích zařízeních různé úrovně i druhu je stěžejní problematika rezervací vzhledem k velkému množství zprostředkovatelů a tzv. incomingových agentur.



Obrázek 33 Primární model – rezervace v ubytovacích zařízeních (fáze 1) (zdroj vlastní)



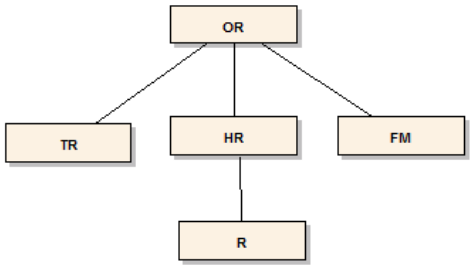
Obrázek 34 Složený model – rezervace v ubytovacích zařízeních (fáze 3 spojená s fází 2) (zdroj vlastní)

V rámci čtvrté fáze pro popsání čtyř principů RBAC bylo identifikováno:

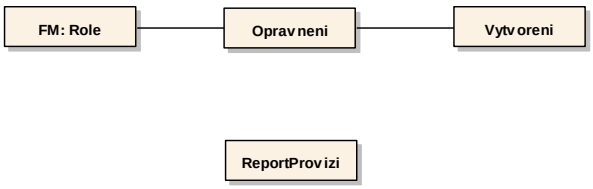
4.3.2.1 Základní jádro

Instance RoleProRezervace:

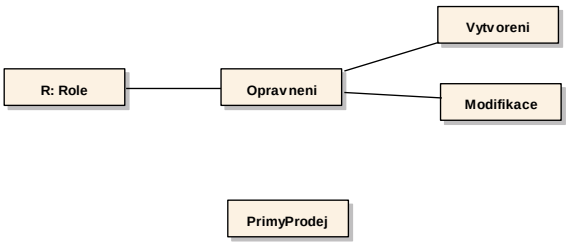
- finanční účetní (FM), referent/ka zajišťující procesně realizaci zakázky (R), hlavní referent/ka (HR), obchodní ředitel/ka (OR) a technický/á ředitel/ka (TR).



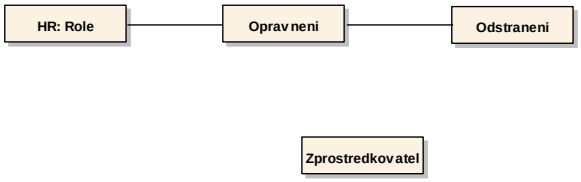
Obrázek 35 Organizační struktura – ubytovací zařízení (zdroj vlastní)



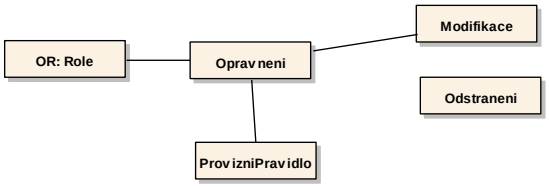
Obrázek 36 Základní jádro RBAC – rezervace v ubytovacích zařízeních (fáze 4 – finanční účetní) (zdroj vlastní)



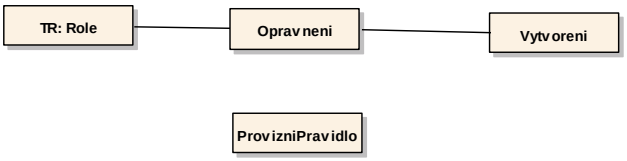
Obrázek 37 Základní jádro RBAC – rezervace v ubytovacích zařízeních (fáze 4 – referent/ka) (zdroj vlastní)



Obrázek 38 Základní jádro RBAC – rezervace v ubytovacích zařízeních (fáze 4 – hl. referent/ka) (zdroj vlastní)

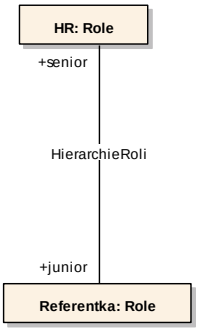


Obrázek 39 Základní jádro RBAC – rezervace v ubytovacích zařízeních (fáze 4 – obchodní ředitel) (zdroj vlastní)

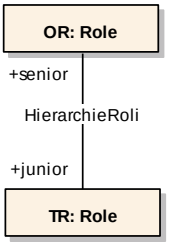


Obrázek 40 Základní jádro RBAC – rezervace v ubytovacích zařízeních (fáze 4 – technický ředitel) (zdroj vlastní)

4.3.2.2 Hierarchie

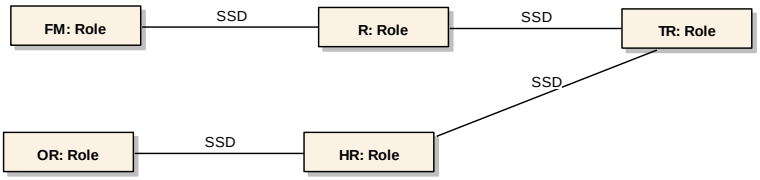


Obrázek 41 Hierarchie RBAC – rezervace v ubytovacích zařízeních (fáze 4 – HR-R) (zdroj vlastní)



Obrázek 42 Hierarchie RBAC – rezervace v ubytovacích zařízeních (fáze 4 – OR-TR) (zdroj vlastní)

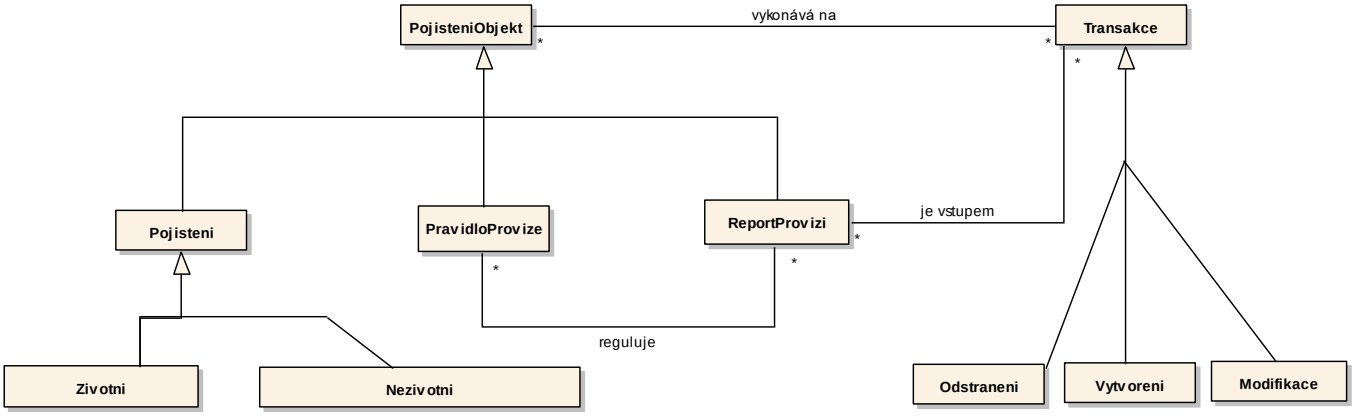
4.3.2.3 SSD, DSD



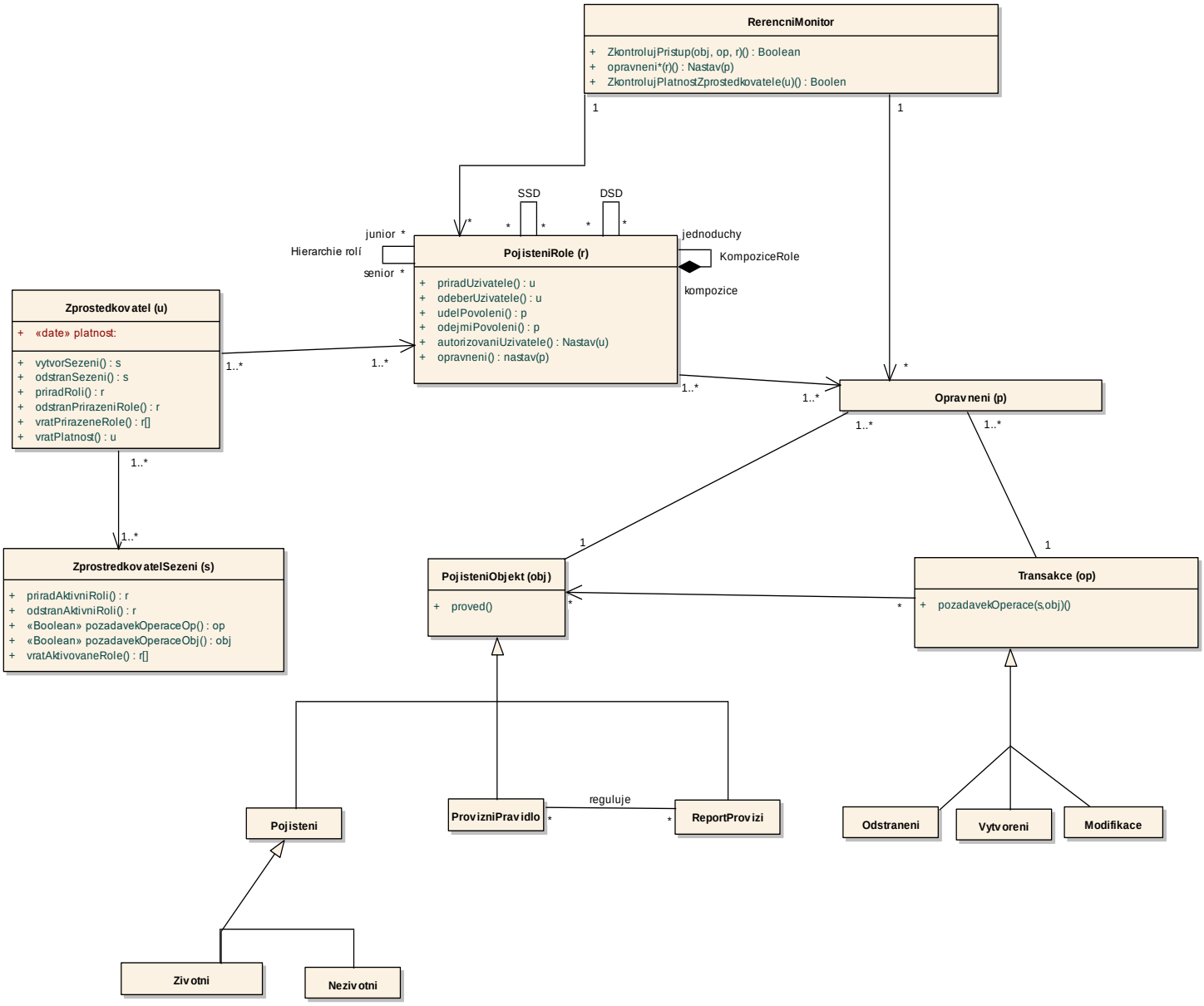
Obrázek 43 SSD, DSD RBAC – rezervace v ubytovacích zařízeních (fáze 4) (zdroj vlastní)

4.3.3 Zprostředkování se zaměřením na pojišťovací sektor

Sektor 65 Pojištění, zajištění a penzijní financování, kromě povinného sociálního zabezpečení je pod dohledem České národní banky (ČNB), velkou část obratu jednotlivým pojišťovnám tvoří obchodní zástupci či zprostředkovatelské firmy, podstatnou skutečnost, které jednotlivé pojišťovací instituce sledují, je platnost zkoušek u ČNB a u nich samotných. Z tohoto důvodu je možné mít platné přístupy, ovšem pokud je jakákoliv zkouška či certifikace neplatná, je přístup odepřen, to je podstatný rozdíl oproti předešlým aplikacím návrhového vzoru.



Obrázek 44 Primární model – pojišťovací sektor (fáze 1) (zdroj vlastní)

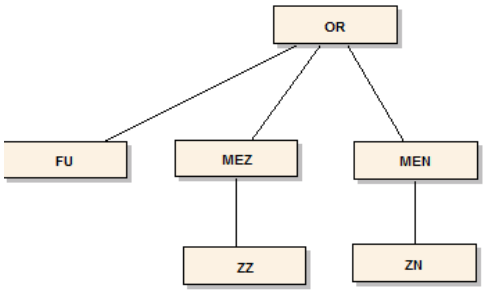


Obrázek 45 Složený model – pojišťovací sektor (fáze 3 spojená s fází 2) (zdroj vlastní)

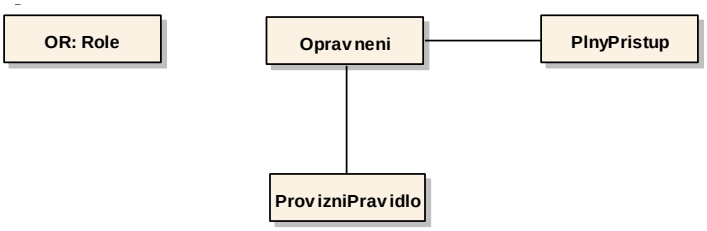
4.3.3.1 Základní jádro

Instance PojisteniRole:

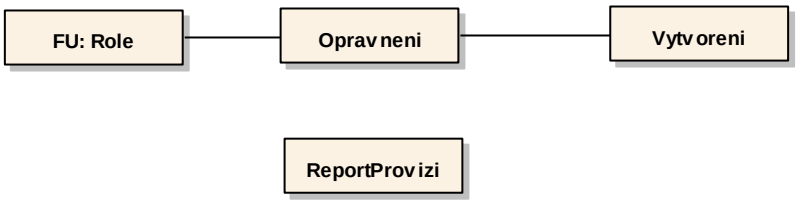
- finanční účetní (FU), zprostředkovatel/ka-obchodní zástupce/kyně životního pojištění (ZZ), zprostředkovatel/ka-obchodní zástupce/kyně neživotního pojištění (ZN), manažer/ka externistů pojišťovny pro životní pojištění (MEZ), manažer/ka externistů pojišťovny pro neživotní pojištění (MEN), obchodní ředitel/ka (OR).



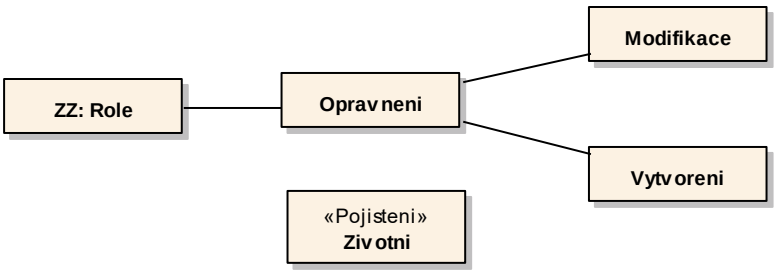
Obrázek 46 Organizační struktura – pojišťovací sektor (zdroj vlastní)



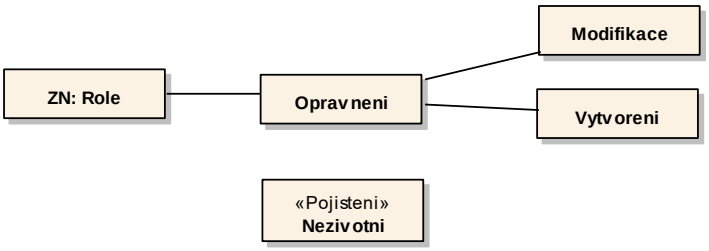
Obrázek 47 Základní jádro RBAC – pojišťovací sektor (fáze 4 – obchodní ředitel/ka) (zdroj vlastní)



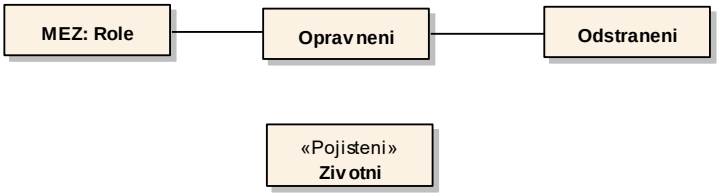
Obrázek 48 Základní jádro RBAC – pojišťovací sektor (fáze 4 – finanční účetní) (zdroj vlastní)



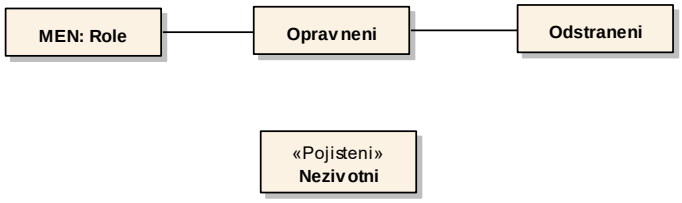
Obrázek 49 Základní jádro RBAC – pojišťovací sektor (fáze 4 – zprostředkovatel/ka život. poj.) (zdroj vlastní)



Obrázek 50 Základní jádro RBAC – pojišťovací sektor (fáze 4 – zprostředkovatel/ka neživ. poj.) (zdroj vlastní)

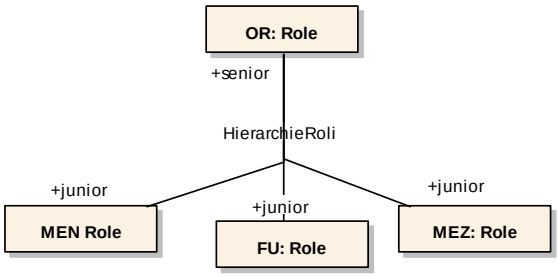


Obrázek 51 Základní jádro RBAC – pojišťovací sektor (fáze 4 – manažer/ka životního poj.) (zdroj vlastní)

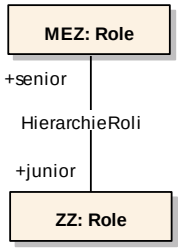


Obrázek 52 Základní jádro RBAC – pojišťovací sektor (fáze 4 – manažer/ka neživotního poj.) (zdroj vlastní)

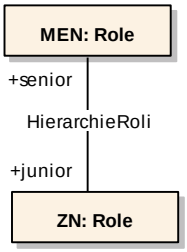
4.3.3.2 Hierarchie



Obrázek 53 Hierarchie RBAC – pojišťovací sektor (fáze 4 – OR-MEN;OR-FU, OR-MEZ) (zdroj vlastní)

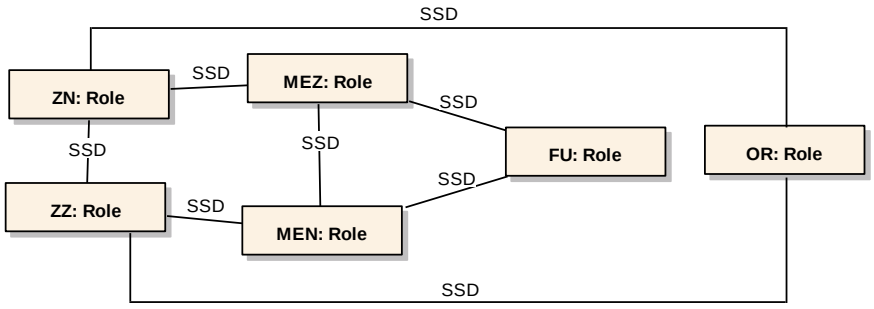


Obrázek 54 Hierarchie RBAC – pojišťovací sektor (fáze 4 – MEZ-ZZ) (zdroj vlastní)



Obrázek 55 Hierarchie RBAC – pojišťovací sektor (fáze 4 – MEN-ZN) (zdroj vlastní)

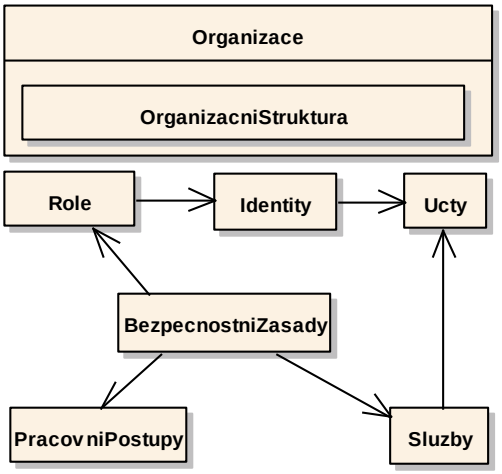
4.3.3.3 SSD, DSD



Obrázek 56 SSD, DSD RBAC – pojišťovací sektor (fáze 4) (zdroj vlastní)

5 Vývoj a inovace v organizaci

Obrázek 57 znázorňuje obecný princip fungování identity managementu. Výchozím bodem jsou bezpečnostní zásady, které jsou aplikovány do služeb a samozřejmě do rolí. Bezpečnostní zásady jsou i výchozím bodem pro stanovení pracovních postupů. Dle služeb vznikají jednotlivé účty, které vlastní identity, tedy konkrétní pracovník organizační jednotky. Role jsou přiřazeny právě konkrétním identitám. Z výše uvedeného vyplývá, že role by měla odpovídat pracovnímu zařazení v organizaci, nebo úrovni oprávnění, případně geografickému členění. Role je tedy mapována na konkrétní přístupové oprávnění – nastavení účtu pomocí bezpečnostní politiky organizace.



Obrázek 57 Princip fungování identity managementu (zdroj vlastní)

5.1 Pracovní náplň uživatele a aktivity v systému

V rámci testování i výchozích testovacích sad rolí se velmi často prokázal nesoulad mezi organizační strukturou a systémovými rolemi, tedy hierarchií organizační a hierarchií rolí. Z této skutečnosti vyplývají zranitelná místa v bezpečnostní politice organizace, která se většinou projevují v případě inovací v organizacích. Z manažerského pohledu se pak jedná o libovolnou organizační změnu, která má dopad na systémové role.

Z terénního výzkumu vyplývá, že konkrétní identity vystupující jako uživatel v roli senior předává svá oprávnění svým junior rolím. Ve většině případů se tak děje na základě skryté zastupitelnosti či přímo fixním přiřazením senior role junior roli. To vede k nesouladu mezi organizační strukturou a systémovými rolemi. Dalším častou bezpečnostní hrozbou jsou samotné zástupy v případě dovolených či nemocenských jednotlivých identit v organizaci a nedodržování časového omezení role. Jsou pochopitelné, ale nikoliv

žádoucí, výjimky, ovšem kontinuální bezpečnostní hrozba v případě řízení přístupů na základě rolí je neakceptovatelná.

Souhrn stěžejních sledovaných parametrů a z nich plynoucí identifikované problémy, které při testování vznikly, jsou zobrazeny v následující tabulce.

Sledovaný parametr	**Identifikované problémy, zranitelná místa**
Odpovídá systémové zařazení organizační struktuře?	Převážně u pracovníků nižších úrovní tedy sekretářek, administrativních pracovníků či pracovníků na nižší úrovni managementu byla nalezena větší práva.
Odpovídají oprávnění skutečnému pracovnímu zařazení?	Překvapivým zjištěním byla trvalá zástupnost v rolích svých nadřízených a tím docházelo k trvalému přístupu k vyšším oprávněním než by odpovídalo skutečnému pracovnímu zařazení daného pracovníka.
Jakým způsobem je řešena odpovědnost jednotlivých pracovníků?	Ve většině organizací docházelo k využívání časového razítka včetně loginu daného uživatele a tím k jednoznačné identifikaci vytvoření, modifikace či odstranění daného procesu či dokumentu.
Jakým způsobem je řešeno řízení uživatelských rolí v případě nasazení nového systému či rozsáhlé inovace v ICT v rámci organizace?	V této situaci dochází velmi často ke konfliktnímu stavu, kdy většina firem potřebuje mít poměrně dlouhou dobu funkční i původní systém a zároveň nový systém, dochází tedy k objemnému sčítání rolí a většinou se v této fázi neřeší duplicita či dokonce triplicita rolí. Tento stav samotní bezpečnostní technici považují za ohrožující.

Tabulka 24 Bezpečnostní hrozby a zranitelná místa v organizacích (zdroj vlastní)

5.2 Změny v organizaci a navržené řešení

Vzhledem k zjištěným nedostatkům a i problémovým oblastím uvedených výše v rámci terénního zkoumání došlo následně k nalezení nedostatků navrženého modelu. Inovace a organizační změny vytvořily další úroveň pohledu na uživatelské přístupy. Převážně je nutné odlišit okamžik, kdy dochází k implementaci výstupního katalogu rolí. Jelikož z přehledu dosažených výsledků vyplývá, že pokud bude navržený mechanismus vycházet pouze z výše uvedených skutečností, pak je vhodný ve dvou situacích:

1. Při realizaci nového systému s cílem dosáhnout nejvýhodnějšího řešení.
2. Aplikace procesu nad existujícím systémem s následnou celkovou přestavbou katalogu rolí, tedy plná aktualizace původního řešení a tím znovu nasazení od počátku.

Z výše uvedeného vyplývá, že navržený postup při hledání ideálních rolí s využitím popsaných algoritmů, neřeší častou situaci z provozní praxe, a to udržení optimálního stavu pro zaběhlý a funkční systém s provozními, organizačními či inovativními změnami. Pokud by bylo zapotřebí provést hledání nového katalogu rolí pro existující systém, který byl již navržen jako nejvhodnější řešení, ale není přizpůsobivý vývoji v rámci organizace, pak vzniká problém, kdy míra optimálního řešení má klesající trend v závislosti na změnách v provozu systému. Katalog rolí v tento okamžik již nelze považovat za nejvhodnější a nalezení opětovné rovnováhy je možné provést pouze tak, že budou respektovány nově vzniklé podmínky provozu systému. Stávající řešení poskytuje jedinou variantu, a to iniciovat kompletní nasazení od počátku. Upravené řešení by přinášelo možnost zaznamenávat tyto změny pomocí začlenění vektoru změn do maximalizačních a minimalizačních úloh a dosáhnout tak přímé reakce na vyvstalé problémy.

Takový vektor změn by:

- doplňoval již vzniklou analýzu,
- upravoval vstupní faktory,
- respektoval poslední hodnoty katalogu rolí.

Přínos případného řešení:

- nevyžaduje kompletní analýzu jako v případě generování nové sady uživatelských rolí,
- zahrnuje a navazuje na předchozí nejvhodnější řešení,

- jedná se o návazný logický krok,
- celková úspora nákladů,
- větší provázanost s aktuálním stavem.

5.3 Pojem změnového vektoru

Jak již bylo zmíněno, pro zachování ideálního stavu katalogu rolí nemusí vždy docházet k opakovanému nasazení z výchozího, dále nulového stavu. Pokud již byla taková forma optimalizace provedena, nabízí se v dalším období na ní navázat, a to s využitím předchozích stavů, do kterých je zahrnut změnový vektor.

Proces hledající ideální řešení lze obecně, bez užití konkrétního algoritmu, definovat jako určité funkcionální maximum (resp. minimum) stavového prostoru S. Matematická formule, vyjadřující tuto operaci je $S´ = f(S)$, kde S je stavový prostor a f optimalizační funkce stavového prostoru.

Pro základní definici použití změnového vektoru lze pak kontinuální optimalizační proces pospat jako $S´´ = f(S´, v)$, kde v je změnový vektor a $S´$ prezentuje stavový prostor z předchozího hledání vhodného katalogu rolí.

Obecně lze takto modifikovanou funkci zajistit pro libovolný optimalizační algoritmus i například využitím Simplexové metody. Vždy je podstatné ohodnotit vhodně prvky změnového vektoru tak, aby výsledné hledání nejvhodnějšího řešení využilo jejich vlastností a v konečné fázi nabídla skutečně ideální řešení nejen v hodnotách, ale i v celkové náročnosti její výpočetní realizace.

Autorka pro přehlednost uvádí, že pojem změnový vektor bude mít v každé konkrétní algoritmizaci zcela jinou matematickou realizaci, přesto tento pojem bude používat jako definiční záležitost.

5.3.1 Řešení pro HILL CLIMB

Z principu prvního použitého algoritmu HILL CLIMB je zřejmá jeho základní nevýhoda, a to uvíznutí v lokálním extrému, navzdory existenci globálního extrému. Řešení tohoto problému je v opakovaném spouštění tohoto algoritmu a konečném vyhodnocení nabídnutých extrémů.

Zavedení změnového vektoru pro tento algoritmus je poměrně jednoduché. Nebude nutné zajišťovat transfer hodnot poslední výstupní sady rolí, ale bude

nutné pouze poskytnout vektor setříděných lokálních extrémů z poslední výstupní sady.

Rozdíl v realizaci je možno formálně zapsat takto:

- První optimalizace
  - $v = (u_{1,} u_{2,} \dots u_{n,})$, kde u jsou postupně všechny vrcholy.
  - $S' = f(S, v)$
- Každé další spuštění
  - $v = (s_{1,} s_{2,} \dots s_{n,} n_{1,} n_{2,} \dots n_{n,})$, kde s jsou postupně všechny vrcholy setříděné v pořadí od globálního extrému až po nejnižší lokální extrém. Vrcholy n jsou pak případné další nové vrcholy, které ještě neprošly předchozí optimalizací.
  - $S' = f(S, v)$

Viditelný problém takového vektoru je zařazení nových prvků na konec setříděného seznamu prvků. Nabízí se varianta přeuspořádání, tzn. provést vyhledávání vhodné varianty pouze na nových vrcholech a pak jej s výslednými vahami zatřídit mezi původní. Toto řešení ovšem nelze pokládat za vhodné, jelikož při použití pouze jakékoliv podmnožiny vrcholů, nebude výsledná sada rolí odpovídat skutečnému řešenému grafu. I proto se původní nástin jeví jako prakticky nejpoužitelnější řešení.

5.3.2 Řešení pro genetický algoritmus

Použití změnového vektoru v oblasti genetických algoritmů bude pracovat s jinými objekty. Zatímco bez použití tohoto vektoru by bylo nutné celý genetický algoritmus aplikovat v plném rozsahu, tzn. postupovat od náhodné populace až k dostatečně ideální a zároveň dohledávat odpovídající fitness funkce, v případě použití změnového vektoru lze celý proces výrazně zjednodušit.

Základní myšlenkou pro zavedení změnového vektoru genetického algoritmu je využití odřezávání starých generací. V prostoru těchto algoritmů se nezachovávají staré informace a vždy se pracuje pouze se vzniklou populací.

Podobně jako v předchozí variantě, je tedy možné využít hodnoty první výstupní sady rolí s tím, že se nebude jednat o vstupní vrcholy, ale o celé populační sady, ke kterým předchozí optimalizační proces dospěl.

Uvedená vlastnost nastiňuje také problém, zda se genetický algoritmus umí vypořádat s vloženou nikoliv nejvhodnější populací. To je situace, která může vzniknout sloučením výsledků předchozí optimalizace a podmínek nového spuštění.

Pro odstranění tohoto problému se nabízejí dvě řešení:

- Využití optimální populace z minulého procesu jako první zdrojové oproti původní, kdy se jednalo o náhodnou. Případné chybějící prvky populace jsou doplněny náhodně.
- Transformace výstupní populace z minulého procesu pomocí mutací a křížení tak, aby se před samotným spuštěním následného hledání nejlepší varianty projevily v původní populaci změny stavového prostoru, které vznikly v období mezi těmito dvěma optimalizacemi.

Z principu genetických algoritmů je zřejmé, že obě varianty dříve nebo později dojdou ke stejnému výsledku.

5.3.3 Řešení pro SAT algoritmus

Problematika SAT algoritmu je co do možností přípravy změnového vektoru nepoměrně bohatší. V každém případě je vhodné analyzovat SAT algoritmy jako takové, a využít možností, které skýtají různá rozšíření a nadstavby.

Jak již bylo v práci popsáno, základním úkolem SAT algoritmu je řešení booleovských formulí. Pro praktickou realizaci se používají tzv. SAT řešiče, kterých je velké množství a každý v sobě skrývá výhody i nevýhody použití. Vzhledem k základní problematice této kapitoly bude následovat srovnání a výběr nejvhodnějšího řešiče pro realizaci opakovaného hledání nejlepšího řešení.

5.3.3.1 WalkSAT

Algoritmus, navržený McAllesterem, Selmanem a Kautzem v roce 1997, je ve svém základním principu navržen na hill-climbing algoritmu (McAllester, et al., 1997). Z pohledu realizace změnového vektoru je tak možno použít stejnou úvahu jako v případě samotného hill-climbingu – stačí místo náhodného počátku zvolit poslední optimální vrchol, resp. globální extrém grafu.

Z principu WalkSAT algoritmu může bohužel vzniknout situace, kdy řešič oznámí, že optimum nemůže najít, ačkoliv existuje – nemůže tedy rozhodnout o nesplnitelnosti formule, ale pouze o její splnitelnosti tím, že najde řešení. Přičte-li se k tomuto řešiči poměrně velká časová náročnost, nebude se jevit pro implementaci změnového vektoru jako ideální.

5.3.3.2 zChaff

zChaff jako SAT řešič implementuje Chaff algoritmus, který se v základní implementaci chová k možnosti použití změnového vektoru podobně jako WalkSAT. Z technického pohledu se jedná o DLL SAT řešič, tedy na rozdíl od DP algoritmů má předvídatelné paměťové nároky a používá VSIDS heuristiku. Ta z principu uchovává skóre každého literálu a přikládá větší váhu naposledy přidaným klauzulím. (Moskewicz et al., 2001)

To, co zChaff nominuje na kandidáta užití změnového vektoru, je jeho technika Two Literal Watching. Ta umožňuje, že na rozdíl od jiných schémat (např. Head/Tail) není třeba během backtrackingu upravovat hlídané literály v databázi klauzulí – to v důsledku znamená, že vyloučíme z hledání klauzule, pokud se jeden z jejích hlídaných literálů stane nepravdivým.

Takto uzavřený stavový prostor se po první optimalizaci stane změnovým vektorem, a to opět ve dvou různých variantách:

- je možné použít stavový prostor vyloučených klauzulí,
- je možné použít stavový prostor nevyloučených klauzulí.

Volba konkrétní varianty bude záviset od očekávaných změn před příštím hledáním ideálního řešení. V případě velkého nárůstu původního prostoru bude vhodné použít prostor s nevyloučenými klauzulemi, protože tím se připraví stavový prostor s podmínkami pro rychlejší implementaci řešiče, než v opačném případě.

Je třeba si uvědomit, že změnový vektor jako pojem v případě použití SAT řešičů je objekt, který funguje v rámci procesu analýzy konfliktu a učení.

Pro tyto účely se používá implikační graf, kdy se definují tzv. jedinečné implikační body UIP. Je to takový uzel, který dominuje uzlům, které se vztahují ke konfliktní

proměnné. zChaff používá ke svému učení algoritmus 1 - UIP, který se experimentálně jeví jako nejrychlejší.

Princip změnového vektoru se tak v prostředí vhodně zvoleného SAT řešiče mění na kognitivní záležitost a jeho základní princip pak dostává zcela jiný rozměr, než je pouhá úprava vstupního prostředí budoucí optimalizace.

5.3.3.3 Shrnutí SAT řešičů

Autorka připomíná, že množství řešičů je veliké a výběr konkrétního nejen ke vztahu k samotné optimalizaci, ale právě k použití změnového vektoru může být znalostně i časově náročným problémem. Zcela jistě by v případě před případové analýzy bylo vhodné věnovat pozornost mezinárodní SAT soutěží (The international sat competitions, 2015), kde lze zjistit podrobné procesní podmínky a výstupy jednotlivých řešičů.

6 Další výzkum

Pro další směřování výzkumu je podstatné hledání řešení pro pokrytí účtů rolemi tak, aby nedošlo k znovu nasazení nového systému rolí, ale pouze k jeho případné modifikaci (vektor změn), kde lze předpokládat, že určité role se stanou fixními. Prvotní analýzu a vstupy popisuje kapitola 5 Vývoj a inovace v organizaci.

Další cestou je hlubší probádání heterogenních systémů, které jsou často využívány převážně v komerční sféře, vyplývající z kapitoly 3 Provozní informační systémy.

Dále je možné aplikovat návrhové vzory RBAC spojené s výstupní sadou katalogu rolí na různé případové studie, převážně na části konkrétních business procesů, vyplývající z kapitoly 4 Návrhové vzory RBAC.

Obecně bylo dále v rámci výzkumu zjištěno několik klíčových oblastí, které je vhodné dále rozvíjet:

- Řešení oblastí v distribuovaných databázích.
- Mobilní a internetové komunikaci (Tomaskova a Simkova, 2013).
- Cloudových řešení.
- NoSQL datatabází.
- NewSQL databází.
- BigData (Hadoop atd.)
- Polyglot databáze.
- Multimodel databáze.
- Řešení systému, kde fungují open data, zde se jednotlivé předešlé oblasti mohou prolínat.

Závěr

Práce se zabývá problematikou bezpečnostního přístupu k informacím. Jejímž cílem je dosažení ideálního definování uživatelských přístupů v informačních systémech. Stěžejní myšlenkou je dosažení co nejmenšího nebo nejvýstižnějšího počtu rolí, které jsou v souladu s potřebami organizace, aby všichni uživatelé měli všechna práva, která skutečně potřebují. Důležitým faktorem při modelování uživatelských rolí je vyvarování se při tvorbě katalogu rolí, nadbytečných práv uživatelů nebo získání náročně řiditelného a nesrozumitelného systému rolí.

Cílem výzkumu bylo provázání existujících systémů pro správu identit, tedy jejich částí týkajících se systému rolí na externí komponentu, která sofistikovaně zajišťuje návrh a životní cyklus systému rolí. Mezi hlavní části patří nejen samotná klasifikace všech potřebných oprávnění přístupů a jejich formalizace pomocí Z-notace, tedy klasifikace přístupů na základě rolí, ale převážně vývoj algoritmů použitelných pro analýzu přístupových oprávnění a modelování uživatelských rolí z konkrétního nastavení oprávnění v koncových systémech.

Proběhl terénní výzkum v několika organizacích, získaná data a podklady umožnily nalezení společných vzorů chování životního cyklu uživatelských přístupů pro jednotlivé ekonomické nebo technicko-provozní činnosti. Tento proces umožnil na základě vstupních parametrů provést analýzy nastavení přístupových oprávnění. Pro ověření zjištěných předpokladů a vzorů chování uživatelských rolí je vytvořena aplikace, která nejen poskytuje ověření nalezených vzorů, ale i predikci optimální varianty, tedy generovat výstupní sady přístupových oprávnění podle předem zadaných kritérií. Aplikace pracuje s rozšířením současné klasifikace a stěžejním faktem, a to provázáním na použité algoritmy. V této části, práce popsala vývoj vhodných algoritmů, které jsou použitelné k modelování katalogu uživatelských rolí, a to algoritmy Hill Climb, SAT-based a genetické algoritmy. Tyto algoritmy byly implementovány nad existující databází uživatelských kont a došlo k jejich následné optimalizaci v reálném prostředí. Algoritmy byly testovány pro existující databáze uživatelských kont v rozsahu 500, 1000 a 2000 kont. V průběhu výzkumu byly stanoveny vhodné vstupní parametry, tak aby odrážely reálné potřeby generování katalogu rolí. Všechny algoritmy byly protestovány pro shodné sady uživatelských kont a stejné parametry řešení, každý algoritmus byl samostatně ohodnocen. V následné fázi došlo k vývoji konektorů na katalogy rolí existujících systémů

pro správu identit. Pro načtení reálných sad uživatelských kont byl otestován a vyvinut konektor pro import uživatelských kont a export rolí.

Výstupní sady rolí byly dále provázány na vybrané návrhové vzory RBAC. Došlo k propojení různých případových studií a následně v kapitole 5 Vývoj a inovace v organizaci byly identifikovány bezpečnostní hrozby, které jsou spojeny s reálnou pracovní náplní uživatele v rámci organizační struktury a jeho skutečných aktivit v systému. Za nejpodstatnější zranitelné místo lze považovat předávání pracovníkům na nižší úrovni oprávnění pracovníků na vyšší úrovni.

Studie organizačních změn a inovací v organizacích přinesla nutnost propojení stávající aplikace pro správu identit s vektorem změn. Zde byl doporučen konkrétní postup pro každý algoritmus zvlášť. Nejrozsáhlejší zkoumání je v případě SAT algoritmu, kdy je nutné rozlišit při využití SAT řešiče zChaff stavový prostor vyloučených klauzulí a nevyloučených klauzulí.

Výše uvedené potvrzuje naplnění hlavního cíle včetně dílčích cílů. Při realizaci hlavního cíle došlo k analyzování přístupových oprávnění a vzorů chování uživatelských rolí. Na základě vybraných algoritmů došlo k formalizaci vstupů s využitím maximalizačních a minimalizačních úloh a realizování testovací aplikace pro generování výstupní sady rolí. Výstupní sady byly použity jako podklad pro aplikaci návrhových vzorů. Testovací fáze aplikace umožnila nalezení zranitelných míst v organizacích, které byly popsány v tabulce 24. V rámci terénního průzkumu cíleného na inovace a vývoj v organizacích došlo k nalezení vektoru změn pro udržitelnost katalogu rolí.

Soupis bibliografických citací

(Ahmed et al., 2012) Ahmed, A., Li R., Bumby, J. Perturbation parameters design for hill climbing MPPT techniques. In: *2012 IEEE International Symposium on Industrial Electronics*. IEEE, 2012, s. 1819-1824.

(Ahn et al., 2000) Ahn, G. J., Sandhu, R., Myong K., Park, J. Injecting RBAC to secure a Web-based workflow system. In: *Proceedings of the fifth ACM workshop on Role-based access control - RBAC '00*. New York, New York, USA: ACM Press, 2000, s. 1-10.

(Ahn a Hu, 2007) Ahn, G. J., Hu, H. Towards realizing a formal RBAC model in real systems. In: *Proceedings of the 12th ACM symposium on Access control models and technologies - SACMAT '07*. New York, New York, USA: ACM Press, 2007, s. 215-224.

(Al-Kahtani et al., 2002) Al-Kahtani, M. A. a Sandhu, R. A model for attribute-based user-role assignment. In: *18th Annual Computer Security Applications Conference, 2002. Proceedings*. IEEE Comput. Soc, 2002, s. 353-362.

(Almutairi et al., 2012) Almutairi, A., Sarfraz, M., Basalamah, S., Aref, W., Ghafoor, A. A distributed access control architecture for cloud computing. *IEEE software*, 2012, 29.2: 36.

(Alturi a Ferraiolo, 2011) Alturi, V., Ferraiolo, D. Role-Based Access Control. In: *Encyclopedia of Cryptography and Security*. Springer US, 2011. s. 1053-1055.

(Barkley et al., 1997) Barkley, J., Cincotta, A., Ferraiolo, D., Gavrila, S., Kuhn, D. R. Role based access control for the world wide web. In: *20th National Computer Security Conference*. 1997. s. 331-340.

(Bertino, 2003) Bertino, E. RBAC models—concepts and trends. *Computers & Security*, 2003, 22.6: 511-514.

(Borghoff et al., 2011) Borghoff, J., Knudsen, L. R., Matusiewicz, K. Hill climbing algorithms and trivium. In: *International Workshop on Selected Areas in Cryptography*. Springer Berlin Heidelberg, 2010. s. 57-73.

(Brunnstein, 2006) Brunnstein, J. *ITIL Security Management realisieren*. Springer Fachmedien, 2006.

(Buckl et al., 2009) Buckl, S., Ernst, A. M., Matthes, F., Ramacher, R., Schweda, C. M. Using enterprise architecture management patterns to complement TOGAF. In: *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*. IEEE, 2009. s. 34-41.

(Cabrera et al., 2016) Cabrera, Armando, Abad, M., Jaramillo, D., Gómez, J., Verdum, J. C. Definition and implementation of the Enterprise Business Layer through a Business Reference Model, using the architecture development method ADM-TOGAF. In: *Trends and Applications in Software Engineering*. Springer International Publishing, 2016. s. 111-121.

(Coyne et al., 2011) Coyne, E. J., Weil, T. R., Kuhn, R. Role engineering: Methods and standards. *IT Professional Magazine, 2011, 13.6: 54*.

(De Haes et al., 2016) De Haes, S., Huygh, T., Joshi, A., Van Grembergen, W. Adoption and Impact of IT Governance and Management Practices: A COBIT 5 Perspective. *International Journal of IT/Business Alignment and Governance (IJITBAG)*, 2016, 7.1: 50-72.

(Doucek a Novotný, 2007) Doucek, P., Novotný, O. Standardy řízení podnikové informatiky. *E+ M Ekonomie a Management/E+ M Economics & Management*, 2007, 2007.3:132.

(Dressler et al., 2010) Dressler, T., Ege, K., Heck, R., Klein, H., Walter, S. M. Collaborative Modelling of ITIL Service Management Processes. *Quality Management for IT Services: Perspectives on Business and Process Performance: Perspectives on Business and Process Performance*, 2010, 125.

(Feltus, Dubois a Petit, 2010) Feltus, C., Dubois, E., Petit, M. Conceptualizing a responsibility based approach for elaborating and verifying RBAC policies conforming with CobiT framework requirements. In: *Requirements Engineering and Law (RELAW), 2010 Third International Workshop on.* IEEE, 2010, s. 34-43.

(Fernandez et al., 2008) Fernandez, E. B., Pernul, G., Larrondo-Petrie, M. M. Patterns and Pattern Diagrams for Access Control. In: *International Conference on Trust, Privacy and Security in Digital Business.* Springer Berlin Heidelberg, 2008. s. 38-47.

(Ferraiolo, 2001) Ferraiolo, D. F. An argument for the role-based access control model. In: *Proceedings of the sixth ACM symposium on Access control models and technologies*. ACM, 2001, s. 142-143.

(Ferraiolo, 2004) Ferraiolo, D. Next generation access control models. In: *Proceedings of the ninth ACM symposium on Access control models and technologies.* ACM, 2004.

(Ferraiolo a Gavrila, 2009) Ferraiolo, D. F., Gavrila, S. I. *Method and system for the specification and enforcement of arbitrary attribute-based access control policies.* U. S. Patent Application No 12/366,855, 2009.

(Ferraiolo a Kuhn, 1992) Ferraiolo, F., Kuhn, D. R, (1992). Role-based access controls. In: *Proceedings of 15th NIST-NCSC National Computer Security Conference.* Baltimore, Maryland: NIST-NCSC, 1992.

(Ferraiolo et al., 1999) Ferraiolo, D. F., Barkley, J. F., Kuhn, D. R. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security (TISSEC)*, 1999, 2.1: 34-64.

(Ferraiolo, et al., 2001) Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., Chandramouli, R. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 2001, 4.3: 224-274.

(Ferraiolo et al., 2003) Ferraiolo, D. F., Chandramouli, R., Ahn, G. J., Gavrila, S. I. The role control center: features and case studies. In: *Proceedings of the eighth ACM symposium on Access control models and technologies.* ACM, 2003, s. 12-20.

(Ferraiolo et al., 2006) Ferraiolo, D., Kuhn, D. R., Chandramouli, R. Role-based access control. Artech House. *Inc., Norwood, MA,* 2006.

(Ferraiolo et al., 2007) Ferraiolo, D., Kuhn, R., Sandhu, R. RBAC standard rationale: Comments on" A critique of the ANSI standard on role-based access control". *IEEE Security & Privacy,* 2007, 5.6: 51-53.

(Ferraiolo et al., 2011) Ferraiolo, D., Atluri, V., Gavrila, S. The Policy Machine: A novel architecture and framework for access control policy specification and enforcement. *Journal of Systems Architecture*, 2011, 57.4: 412-424.

(Flanagin a Metzger, 2007) Flanagin, A. J., Metzger, M. J. The role of site features, user attributes, and information verification behaviors on the perceived credibility of web-based information. *New Media & Society*, 2007, 9.2: 319-342.

(Hochstein et al., 2005) Hochstein, A., Zarnekow, R., Brenner, W. ITIL as common practice reference model for IT service management: formal assessment and implications for

practice. In: *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*. IEEE, 2005. s. 704-710.

(Gollmann, 2010) Gollmann, D. Computer Security. *Wiley Interdisciplinary Reviews: Computational Statistic,* 2010, 2.5: 544-554.

(Hu a Kent, 2012) Hu, V. C., Kent, K. A. *Guidelines for access control system evaluation metrics*. US Department of Commerce, National Institute of Standards and Technology, 2012.

(Hu et al., 1993) Hu, M. Y., Demurjian, S. A., Ting, T. C. User-role based security profile for an object-oriented design model. In: *Results of the Sixth Working Conference of IFIP Working Group 11.3 on Database Security on Database security, VI: status and prospects: status and prospects.* Elsevier Science Inc., 1993. s. 333-348.

(Hu et al., 2001) Hu, V. C., Frincke, D. A., & Ferraiolo, D. F. The policy machine for security policy management. In: *International Conference on Computational Science-ICCS 2001*. Springer Berlin Heidelberg, 2001. s. 494-503.

(Hu et al., 2006) Hu, V. C., Ferraiolo, D., Kuhn, D. R. *Assessment of access control systems*. US Department of Commerce, National Institute of Standards and Technology, 2006.

(Hu et al., 2006, June) Hu, V. C., Kuhn, D. R., Ferraiolo, D. F. The computational complexity of enforceability validation for generic access control rules. In: *IEEE-International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC´06).* IEEE 2006. s. 260-267.

(Hu et al., 2007) Hu, V. C., Ferraiolo, D. F., Scarfone, K. Access control policy combinations for the grid using the policy machine. In: *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid´07).* IEEE, 2007. s. 225-232.

(Hu et al., 2008) Hu, V., Quirolgico, S., Scarfone, K. Access Control Policy Composition for Resource Federation Networks Using Semantic Web and Resource Description Framework (RDF). 2008.

(Hu et al., 2008, December) Hu, V. C., Kuhn, D. R., Xie, T. Property verification for generic access control models. In: *Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on.* IEEE, 2008. s. 243-250.

(Hu et al., 2011) Hu, V. C., Kuhn, D. R., Xie, T., Hwang, J. Model checking for verification of mandatory access control models and properties. *International Journal of Software Engineering and Knowledge Engineering*, 2011, 21.01: 103-127.

(Hu et al., 2014) Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K. Guide to attribute based access control (ABAC) definition and considerations. *NIST Special Publication*, 2014.

(Hu et al., 2015) Hu, V. C., Kuhn, D. R., Ferraiolo, D. F. Attribute-Based Access Control. *IEEE Computer*, 2015, 48.2: 85-88.

(INCITS, 2012) INCITS, ANSI. INCITS 359-2012, American national standard for information technology, Information Technology - Role Based Access Control. . *American National Standards Institute*, 2012.

(INCITS, 2004) INCITS, ANSI. INCITS 359-2004, American national standard for information technology, role based access control. *American National Standards Institute*, 2004.

(ISO, 2016) *International Organization for Standardization ISO Central Secretariat* (online). Ženeva, Švýcarsko: International Organization for Standardization ISO Central Secretariat, 2016 (cit. 2016-07-03). Dostupné z: http://www.iso.org.

(Kern et al., 2004) Kern, A., Kuhlmann, M., Kuropka, R., Ruthert, A. A meta model for authorisations in application security systems and their integration into RBAC administration. In: *Proceedings of the ninth ACM symposium on Access control models and technologies*. ACM, 2004. s. 87-96.

(Kim et al., 2004) Kim, D-K., Ray, I., France, R., Li, N. Modeling role-based access control using parameterized UML models. In: *International Conference on Fundamental* Approaches to Software Engineering. Springer Berlin Heidelberg, 2004. s. 180-193.

(Kim et al., 2006) Kim, D-K., Mehta, P., Gokhale, P. Describing access control models as design patterns using roles. In: *Proceedings of the 2006 conference on Pattern languages of programs.* ACM, 2006. s. 11.

(Klarl et al. 2009) Klarl, H., Molitorisz, K., Emig, C., Klinger, K., Abeck, S. Extending Role-based Access Control for Business Usage. In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE, 2009. s. 136-141.

(Krutz et al., 2010) Krutz, R. L.; Vines, R. D. *Cloud security: A comprehensive guide to secure cloud computing*. Wiley Publishing, 2010.

(Kuhn, 2011) Kuhn, D. R. Vulnerability hierarchies in access control configurations. In: *Configuration Analytics and Automation (SAFECONFIG), 2011 4th Symposium on.* IEEE, 2011. s. 1-9.

(Kuhn et al., 2010) Kuhn, D. R., Coyne, E. J., Weil, T. R. Adding attributes to role-based access control. *IEEE Computer*, 2010, 43.6: 79-81.

(Latham, 1986) Department of Defense trusted computer system evaluation criteria. *Department of Defense*, 1986, DoD 5200.28-STD.

(Mahdavi et al., 2003) Mahdavi, K., Harman, M., Hierons, R. M. A multiple hill climbing approach to software module clustering. In: *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on.* IEEE, 2003. s. 315-324.

(McAllester, et al., 1997) McAllester, D., Selman, B., and Kautz, H. Evidence for invariants in local search. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, 1997. s. 321–326.

(Moskewicz et al., 2001) Moskewicz, M., W., Madigan, C. F., Zhao, Y., Lintao, Z., and Malik, S., Chaff: Engineering an efficient SAT solver. In: *Proceedings of the 38th annual Design Automation Conference.* ACM, 2001. s. 530-535.

(Mosse, 2002) Mossé, F. G. Modeling Roles - A Practical Series of Analysis Patterns.*Journal of Object Technology*, 2002, 1.4: 27-37.

(Moon et al., 2004) Moon, C. J., Park, D. H., Park, S. J.,Baik, D. K. Symmetric RBAC model that takes the separation of duty and role hierarchies into consideration. *Computers & Security*, 2004, 23.2: 126-136.

(Ni et al., 2009) Ni, Q., Bertino, E., Lobo, J., Calo, S. B. Privacy-aware role-based access control. *IEEE Security & Privacy*, 2009, 4.7: 35-43.

(O'Connor a Loomis, 2010) O'Connor, A. C., Loomis, R. J. 2010 Economic Analysis of Role-Based Access Control. *NIST, Gaithersburg, MD*, 2010, 20899.

(Park, 2001) Park, J. S., Sandhu, R., Ahn, G. J. Role-based access control on the web. *ACM Transactions on Information and System Security (TISSEC)*, 2001, 4.1: 37-71.

(Portela, 2010) Portela, I. M. (ed.). *Information Communication Technology Law, Protection and Access Rights: Global Approaches and Issues: Global Approaches and Issues.* IGI Global, 2010.

(Rajpoot et al., 2015) Rajpoot, Q. M., Jensen, C. D., Krishnan, R. Integrating attributes into role-based access control. In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer International Publishing, 2015. s. 242-249.

(Rashid et al., 2010) Rashid, Z., Basit, A., Anwar, Z. TRDBAC: Temporal reflective database access control. In: *Emerging Technologies (ICET), 2010 6th International Conference on. IEEE*, 2010. s. 337-342.

(Ray et al., 2004) Ray, I., Li, N., France, R., Kim, D. K. Using UML to visualize role-based access control constraints. In: *Proceedings of the ninth ACM symposium on Access control models and technologies.* ACM, 2004. s. 115-124.

(Sandhu et al., 1996) Sandhu, R. S., Coyne, E. J., Feinstein, H. L., Youman, C. E. Role-based access control models. *IEEE Computer*, 1996, 29.2:38-47.

(Sandhu et al., 2000) Sandhu, R., Ferraiolo, D., Kuhn, R. The NIST model for role-based access control: towards a unified standard. In: *ACM workshop on Role-based access control* 2000.

(Sheikhpour a Modiri, 2012) Sheikhpour, R., Modiri, N. A best practice approach for integration of ITIL and ISO/IEC 27001 services for information security management. *Indian Journal of Science and Technology*, 2012, 5.2: 2170-2176.

(Simkova, 2013) Simkova, M. Boolean SATisfiability problem (SAT) in modelling user roles. In: *Global Journal on Technology,* 2013, 4.2: 1022-1028.

(Simkova a Poulova, 2012) Simkova, M., Poulova, P. (2012). The system of user roles and modeling of access privileges. In: *International Conference on COMPUTERS (ICCOMP12),* 2012, s. 382-385.

(Simkova a Stepanek, 2012) Simkova, M., Stepanek, J. The optimal setting of access rights by generating output set of access roles . In: *International Conference on DATA NETWORKS, COMMUNICATIONS, COMPUTERS (DNCOCO '12),* 2012, s. 173-176.

(Simkova a Tomaskova, 2012) Simkova M., Tomaskova, H. Application of max-min algebra for modeling of system of user roles. In: *International Conference on MATHEMATICAL and*

COMPUTATIONAL METHODS in SCIENCE and ENGINEERING (MACMESE'12), 2012, s. 270-274.

(Simkova a Tomaskova, 2013) Simkova, M., Tomaskova, H. Modeling of user roles for mobile communication in fuzzy algebra. In: *Global Journal on Technology*, 2013, 4:2: 1029-1034.

(Shin, 2000) Shin, M. E., Ahn, G. J. UML-based representation of role-based access control. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.(WET ICE 2000). Proeedings. IEEE 9th International Workshops on*. IEEE, 2000, s. 195-200.

(Stepanek a Simkova, 2012) Štěpánek, J., Šimková, M. Modeling roles – description of used algorithms in the system for the analysis of setting of access permissions. In: *International Conference on DATA NETWORKS, COMMUNICATIONS, COMPUTERS (DNCOCO '12)*, 2012, s. 188-191.

(Strembeck a Mendling, 2011) Strembeck, M.; Mendling, J. Modeling process-related RBAC models with extended UML activity models. *Information and Software Technology*, 2011, 53.5: 456-483.

(The international sat competitions, 2015) *The international sat competitions* (online). Arras, Francie: Innovation and Enterprise Research Laboratory, University of Technology, Sydney a Université d'Artois, Francie, 2015 (cit. 2016-07-20). Dostupné z: http://www.satcompetition.org/.

(Tomaskova a Simkova, 2013) Tomaskova, H., Simkova, M. (2013). Use of Modelling Roles in Internet and Mobile Communications. *Global Journal on Technology*, 2013.3: *910-914*.

(Younis et al., 2014) Younis, A., Kifayat, K., Merabti, M. An access control model for cloud computing. *Journal of Information Security and Applications*, 2014, 19.1: 45-60.

(Yu a Brewster, 2012) Yu, S., Brewster, J. Formal specification and impementation of RBAC model with SOD. *Journal of Software*, 2012, 7.4: 870-877.

(Yue a Jian, 2011) Yue, L., Jian, L. ITIL Application Analysis in the Process Optimization of Reference Work (J). *Library Journal*, 2011, 2: 006.

(Zelenka a Borkovcova, 2016) Zelenka, J., Borkovcova, M. The Use of Object-Oriented Integrated Information Systems in E-learning. *Advanced Science Letter*, 2016, *přijatý*.

Seznam autorských publikací

Borkovcová, M.: Použitelnost e-shopů. In: *Sborník příspěvků IMEA 2014*, 2014.

Borkovcová, M., Štěpánek, J.: Architecture of Low Cost Data Centers - practical possibility. In: *International-business-Information-Management-Association Conference.* 2014. s. 2746-2754.

Borkovcová, M., Grulich, P.: Web Applications Security in the Field of Archiving. In: *International-business-Information-Management-Association Conference.* 2015, s. 2746-2754.

Borkovcová, M., Borkovec, R.: Effective Interconnection UML and Realized Applications "Use of UML in Small Companies and Development Teams. In: *International-business-Information-Management-Association Conference.* 2014, s. 1884-1887.

Borkovcová, M.: Architecture of Low Cost Data Centers. In: *International-business-Information-Management-Association Conference.* 2014, s. 1319-1322.

Borkovcová, M., Štěpánek, J.: Designing Tracking System as an Online Service using HTML5 GeoAPI. In: *International-business-Information-Management-Association Conference.* 2014, s. 1147-1150.

Borkovcová, M., Bureš, V.: Conceptual model of software solution for transport logistics in the field of municipal services. In: *International-business-Information-Management-Association Conference.* 2014, s. 1874-1883.

El-Hmoudová, D., Borkovcová, M.: Educational Program BigData EduCloud at Faculty of Informatics and Management. In: *International Conference on Web-Based Learning.* Springer Berlin, 2015, s. 49-58.

Grulich, P., Borkovcová, M.: Digitizing and Electronic Documents in Archiving. In: *International-business-Information-Management-Association Conference.* 2015, s. 2837-2844.

Grulich, P., Borkovcová, M.: Continuous Care of a Digital Archive - Theses. In: *International-business-Information-Management-Association Conference.* 2015, s. 2868-2876.

Němcová, Z., Šimková, M., Clouba, T.: Application for education of financial literacy. In: *Procedia-Social and Behavioral Sciences,* 2011, 28:370-373.

Otčenášková, T., Kolerová, K., Němcová, Z. Borkovcová, M., Cimler, R., Stropková, A. Bureš, V.: Softwarové aplikace pro vizualizace v automatizaci In: *Sborník příspěvků IMEA 2014*, 2014, s.218-224.

Poulová, P., Šimonová, I., Šimková, M.: Database System within the Computer Science Education. In: *Recent Researches in Communications and Computers*. 2012, s. 315-320.

Šimková, M., Němcová, Z., Tomášková, H.: Mobile education in tools. In: *Procedia-Social and Behavioral Sciences*,.2012, 47: 10-13.

Šimková, M., Štěpánek, J.: Effective use of virtual learning environment and LMS. In: *International-business-Information-Management-Association Conference.* 2013, s. 497-500.

Šimková, M., Štěpánek, J., Cimler, R.: University with mobile approach ICT. In: *Procedia-Social and Behavioral Sciences*, 2012, 47: 79-82.

Šimková, M.: Using of Computer Games in Supporting Education. In: *International-business-Information-Management-Association Conference*. 2014, s. 1224-1227.

Šimková, M., Tomášková, H.: Modeling of user roles for mobile communication in fuzzy algebra. In: *Global Journal on technology*. 2013, 2013.4: 1029-1034.

Šimková, M., Štěpánek, J.: Design and implementation of simple interactive e-learning system. In: *Procedia-Social and Behavioral Scinences*. 2013, 83: 413-416.

Šimková, M.: Boolean SATisfiability problem (SAT) in modelling user roles. . In: *Global Journal on technology*. 2013, 2013.4: 1022-1028.

Šimková, M., Poulová, P.: The System of User Roles and Modelling of Access Privileges. In: *Recent Researches in Communications and Computers*. 2012, s.382-385.

Šimková, M., Štěpánek, J.: Modeling roles - description of used algorithms in the system for the analysis of setting of access permissions. In: *Advances in Data Networks, Communications, Computers and Materials*. 2012, s.188-191.

Šimková, M., Štěpánek, J.: Comparing web pages in terms of inner structure. In: *Procedia-Social and Behavioral Scinences*. 2013, 83: 458-462.

Šimková, M., Štěpánek, J.: Usage of IT to support teaching in the financial education program. In: *Procedia-Social and Behavioral Scinences*. 2013, 83: 454-457.

Šimková, M.: Procesy organizační změny vyvolané aplikací mobilních prostředků ICT. In: *Hradecké ekonomické dny 2011*. 2011, s. 338-342.

Šimková, M., Poulová, P.: Identification of critical places of database machine with using of expert rules. In: *Procedia information technology and computer science.* 2012.3: 815-820.

Šimková, M., Štěpánek, J.: The optimal setting of access rights by generating output set of access roles. In: *Advances in Data Networks, Communications, Computers and Materials*. 2012, s. 173-176.

Šimková, M., Poulová, P.: Performance analysis of the database. In: *Procedia information technology and computer science.* 2012.3: 802-807.

Štěpánek, J., Šimková, M., Cimler, R.: Affect of Watched Surroundings on Agent Decision Making on Stock Market. In: *Procedia-Social and Behavioral Sciences*, 2012, 47: 14-18.

Tomášková, H., Šimková, M.: Application of max-min algebra for modeling of system of user roles. In: *Mathematiccal and computational methods in sciences and engineering*, 2012, s. 270-274.

Tomášková, H., Němcová, Z., Šimková, M.: Usage of Virtual Communication in University Environment. In: Procedia-social and behavioral sciences, 2011, 48: 360-364.

Tomášková, H., Šimková, M.: Use of modelling roles in internet and mobile communications. In: *Global Journal on Technology,* 2013, 4.2: 910-914.

Přijaté publikace, čekající na zveřejnění

(Horalek et al., 2016) Horalek, J., Holik, F., Borkovcova, M., Svoboda, T. Using Rasberry Pi to Optimize Audio Output. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 2016, *přijatý*.

(Zelenka a Borkovcova, 2016) Zelenka, J., Borkovcova, M. The Use of Object-Oriented Integrated Information Systems in E-learning. *Advanced Science Letter*, 2016, *přijatý*.

Seznam příloh

Příloha 1 – Nástroj pro modelování rolí

Specifikace systému na modelování rolí

Systém, který na základě vstupních parametrů provádí analýzu nastavení přístupových oprávnění a generuje výstupní sestavy rolí. Systém načte účty z LDIF nebo LDAP jednotného formátu - všechny musí mít stejný počet a typ konstantních atributů. Uživatel si vybere atributy, které budou klíčové pro generování rolí, a nadefinuje pravidla pro sčítání rolí.

Poté položí jeden z následujících dotazů:

- Najít X rolí, aby pokryly maximum účtů (např. najít 1 základní roli, jenž pokryje skoro všechny účty).
- Najít co nejméně rolí, které pokryjí alespoň X% účtů (většinou 95% nebo 98%).
- Fixace rolí na zvolený atribut. Počet rolí bude právě roven nebo vyšší než je doména tohoto atributu. Analýza rolí tedy bude probíhat s tím, že se budeme zajímat o kombinace zbývajících atributů.

Problém, který se řeší při dotazování, je podobný problému hledání optimálního rozvrhu nebo cesty. Poté co systém najde požadované možnosti kombinací rolí, má uživatel možnost preview na změny v modelu. Samozřejmostí je export případně import rolí, či filtrace účtů. Uživatel má tak přehled, které účty nejsou pokryté a proč, procentuální statistiky ohledně podílení se na pokrytí účtů pomocí jednotlivých rolí. Dále lze zjistit, jak se situace změní při dodefinování nové role, změně definice nějaké role, odebraní role.

LDAP

LDAP (Lightweight Directory Access Protocol) je definovaný protokol pro ukládání a přístup k datům na adresářovém serveru. Podle tohoto protokolu jsou jednotlivé položky na serveru ukládány formou záznamů a uspořádány do stromové struktury (jako ve skutečné adresářové architektuře). Je vhodný pro udržování adresářů a práci s informacemi o uživatelích (např. pro vyhledávání adres konkrétních uživatelů v příslušných adresářích, resp. databázích). LDAP aplikace funguje na bázi klient-server. V komunikaci využívá jak synchronní tak asynchronní mód. Součástí LDAP je autentizace klienta.

Data jsou uchovávána ve stromové struktuře pomocí záznamů. Pod pojmem záznam si lze představit souhrn atributů (dvojice jméno - hodnota). Každý záznam musí být jednoznačně identifikovatelný pomocí svého rozlišovacího jména (DN = Distinguished Name) v rámci celého stromu serveru. Atributy nesou informaci o stavu daného záznamu. Záznamy, uložené v adresáři, musí odpovídat přesně definovanému schématu. Schéma tvoří soubor povolených objektových tříd a k nim náležících atributů. Z faktu, že každý záznam je instancí objektové třídy, vyplývá, že musí obsahovat všechny atributy vedené u dané objektové třídy jako povinné. Mimo to může záznam obsahovat i atributy nepovinné, nicméně opět musí vybírat pouze z množiny příslušející dané objektové třídě.

LDIF

LDIF je standardizovaný formát pro reprezentaci dat na adresářovém serveru. Jedná se o jednoduchou textovou reprezentaci záznamů v adresáři. Každý záznam je v něm definován svým rozlišovacím jménem DN a povinnými (či nepovinnými) atributy. U každého záznamu je nutné uvést, jaké objektové třídy, popřípadě tříd, je záznam instancí.

CSV

CSV (Comma-separated values) je jednoduchý souborový formát určený pro výměnu tabulkových dat. Soubor ve formátu CSV sestává z řádků, ve kterých jsou jednotlivé položky odděleny znakem čárka (,). Hodnoty položek mohou být uzavřeny do uvozovek ("), což umožňuje, aby text položky obsahoval čárku. Pokud text položky obsahuje uvozovky, jsou tyto zdvojeny.

Požadavky na hardware

Pro fungování podpůrného nástroje Role Modeler jsou minimální nikoliv optimální požadavky následující - počítač s procesorem 2GHz a 1024MB RAM operační paměti. V případě použití algoritmu SAT-based, jsou nároky na paměť výrazně vyšší. Jádrem tohoto algoritmu je řešič problému SAT. Řešiči se musí předat formule kompletně popisující problém k vyřešení a ta u většího počtu účtů nebo hledaných rolí může enormně narůst. Při modelování rolí nad mnoha tisíci účty může být potřeba až 4GB RAM operační paměti, pro využití SAT algoritmu je doporučeno výkonnější počítačové zařízení.

Požadavky na software

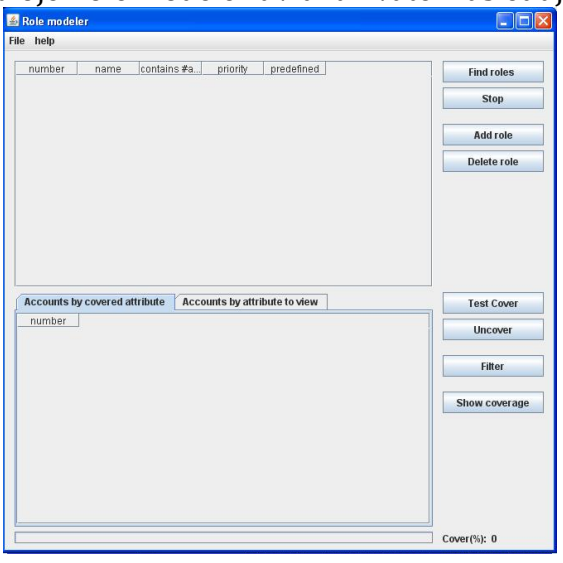
Nástroj Role modeler nepoužívá žádné systémově závislé komponenty. Pro jeho provoz stačí prostředí JRE (Java Runtime Enviroment), které lze zdarma stáhnout ze stránek Oracle.

Instalace spočívá ve zkopírování složky RoleModeler na místo na disku, z kterého si uživatel přeje program používat.

- Program se spouští v adresáři souborem run.bat, kde se nachází soubor Rolemodeler.jar.
- Pokud uživatel potřebuje z důvodu používání SAT-based algoritmu zvýšit paměť, která je k dispozici pro prostředí, v němž program Role modeler běží, může to udělat např. spuštěním programu Role modeler z příkazové řádky následujícím příkazem:
    - java – jar -XmsYYYm –XmxZZZm Rolemodeler.jar, kde se za YYY dosadí paměť v MB, kterou prostředí dostane při spuštění programu, a za ZZZ se dosadí velikost paměti v MB, kterou má prostředí pro běh programu Role modeler maximálně k dispozici.
- Zkušební LDIF soubory s účty jsou k dispozici ve složce RoleModeler/ldif, např.:
    - acc_100.ldif obsahuje 100 účtů,
    - acc_500.ldif obsahuje 500 účtů.

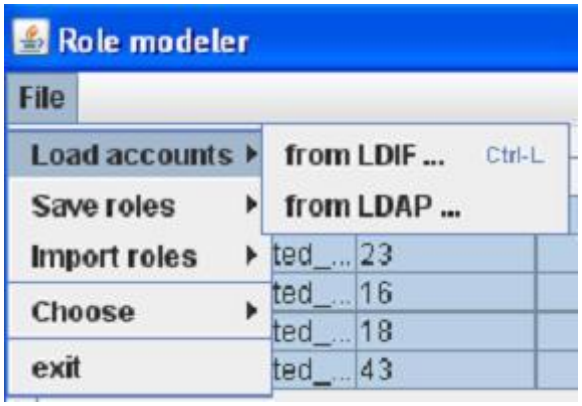
První kroky po spuštění Role modeleru

Po prvním spuštění nástroje Role modeler uvidí uživatel následující okno.



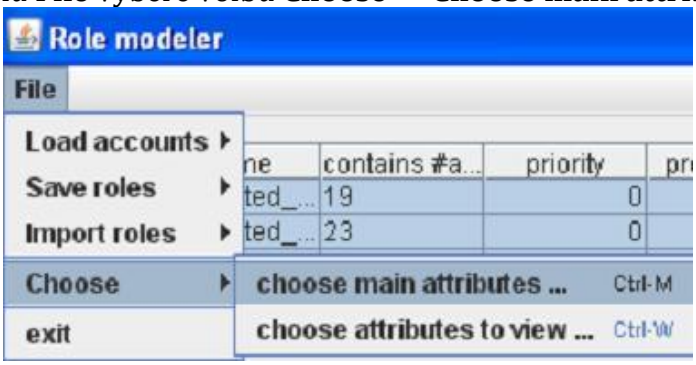
Příloha - Obrázek 1 První spuštění (zdroj vlastní)

Nejdříve je nutné načíst účty a potom určit, nad kterými atributy se budou modelovat role tedy hlavní atributy. Účty lze načíst pomocí **Load accounts** v menu **File**, nebo pomocí klávesové zkratky **CTRL+L**, která uživatele přesune do souborového vyhledávače **LDIF** souborů.



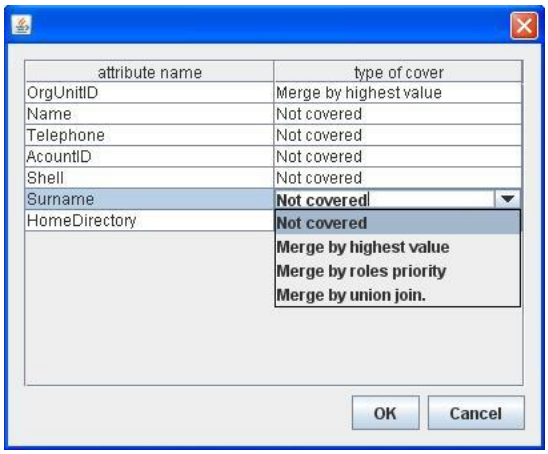
Příloha - Obrázek 2 První spuštění - load accounts (zdroj vlastní)

Poté uživatel v menu **File** vybere volbu **Choose -> Choose main attributes....**



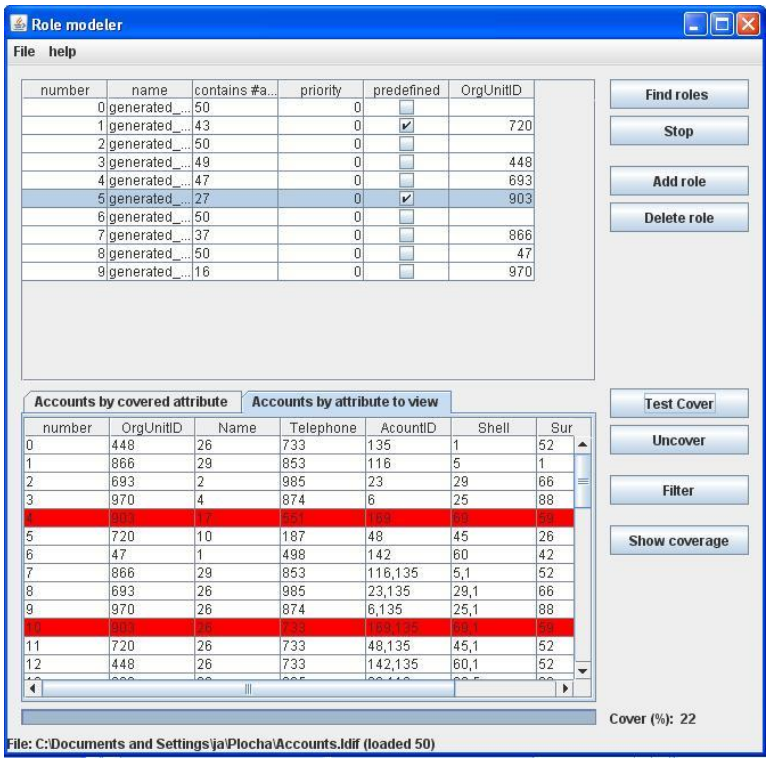
Příloha - Obrázek 3 První spuštění – choose main attributes (zdroj vlastní)

Zde si uživatel určí hlavní atributy tím způsobem, že u požadovaného atributu klikne na políčko **type of cover**, čímž si rozbalí roletu s možnostmi. U vybraných hlavních atributů si uživatel zvolí typ definice, u ostatních atributů zůstane hodnota **Not covered**.



Příloha - Obrázek 4 První spuštění – types of cover (zdroj vlastní)

Nyní si už může uživatel pomocí tlačítka **Find roles** nechat dle svých požadavků generovat skupiny rolí. Úkol je možné usnadnit například tím, že si uživatel předtím pomocí **Filter** profiltruje účty na generování rolí.



Příloha - Obrázek 5 Hlavní okno (zdroj vlastní)

Hlavní okno se skládá ze dvou tabulek, které jsou umístěny nad sebou. Každá z nich je vpravo doplněna skupinou tlačítek. Pod spodní tabulkou se nachází lišta, na které je vidět, zda program právě vykonává nějakou náročnější operaci. Napravo od lišty je údaj udávající pokrok při modelování rolí. Pokud se algoritmus snaží nalézt maximální pokrytí, ukazuje, kolik procent účtů se už pomocí nalezených rolí podařilo pokrýt. Když se algoritmus snaží nalézt minimum rolí, ukazuje, kolik rolí už bylo nalezeno.

Horní tabulka v hlavním okně se nazývá tabulka rolí. Obsahuje role, které si uživatel vygeneroval, importoval nebo ručně přidal. Kromě jiných sloupců obsahuje sloupce s hlavními atributy, což jsou atributy, nad kterými se tvoří role. N**umber** označuje pořadové číslo role, **name** odpovídá jménu role a jedná se o editovatelné pole. **Contains accounts** udává procento účtů, u nichž daná role odpovídá pokrytí, **priority** značí prioritu role a jedná se o editovatelné pole. Pokud je zaškrtnuto **predefined**, bude tato role použita při následujícím modelování, zaškrtnutím tohoto políčka uživatel vlastně říká, že je to „uživatelova role“ a lze potom u ní modifikovat hlavní atributy. Následují hlavní atributy,

na obrázku 5 je jenom jeden, a to OrgUnitID. Pomocí tlačítek **Add role**/**Delete role** lze přidávat/odebírat role. Přidaná role, na rozdíl od role vygenerované, neobsahuje žádné hodnoty v hlavních atributech.

Chce-li uživatel vybrat souvislou skupinu řádků v tabulce, tedy řádky, které leží hned pod sebou, stačí jednou kliknout levým tlačítkem myši a táhnout nahoru nebo dolů. V případě výběru všech řádků v tabulce, označí kterýkoli z řádků a stiskne CTRL+A. Vybírání nesouvislé skupiny řádků v tabulce lze pomocí klávesy CTRL + klik levým tlačítkem myši na dané řádky, tímto způsobem se řádky nejen vybírají, ale je možné je i odebrat. Pro případ odznačit již označený řádek v tabulce lze využít CTRL + klik levým tlačítkem myši na daný řádek. Pro setřídění tabulky dle hodnot v některém ze sloupců, provede klik levým tlačítkem myši na nadpis daného řádku. Změna pořadí sloupců v tabulce se provádí pomocí kliku levým tlačítkem myši na nadpis řádku a táhnutím doleva nebo doprava.

Spodní tabulka v hlavním okně se nazývá tabulka účtů. Obsahuje dvě záložky: **Accounts by atribute to view**, tedy tabulka viditelných atributů a **Accounts by covered atribute** označující tabulku hlavních atributů. Výběr zobrazení atributů, viditelné, nebo hlavní atributy, se specifikuje v menu **file->choose-> v horní tabulce**. Obě záložky tabulky účtů obsahují načtené účty. Pokud je použit filtr, obsahují tabulky v záložkách jen účty, které byly profiltrovány.

Tlačítka

Význam a funkce jednotlivých tlačítek v hlavním okně v pořadí odshora dolů:

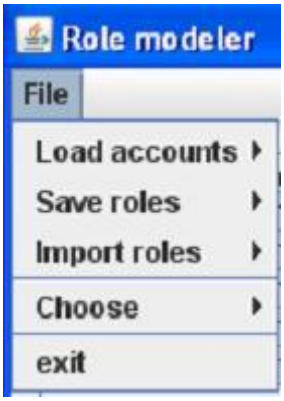
- **Find roles** – vyvolá okno pro modelování rolí, předpokladem pro modelování je předchozí určení hlavních atributů.
- **Stop** – přeruší práci algoritmu, který právě modeluje role, při použití SAT řešiče je reakce závislá na nastavení algoritmu před zahájením.
- **Add role** – přidá roli do tabulky rolí, u přidané role jsou hlavní atributy prázdné, priorita je rovna 0, je implicitně zaškrtnuté políčko predefined, podmínkou pro hledání rolí je určení hlavních atributů.
- **Delete role** – odstraní označenou roli/označené role z tabulky rolí.
- **Test cover** – ukazuje pokrytí rolemi na účtech v tabulce účtů. Pokud v tabulce rolí nejsou žádné role označené, vybarví se žlutě účty, které jsou pokryty všemi rolemi.

Pokud v tabulce rolí jsou označené role, ukáže program červeně účty, které jsou těmito rolemi pokryty. Má-li být nějaký účet vybarven červeně i žlutě zároveň, střídá se více testů pokrytí za sebou, zbarví se tento účet oranžově.

- **Uncover** – „odbarví“ se tabulka účtů, přestane ukazovat pokrytí rolemi na účtech.
- **Filter** – vyvolá okno pro filtrování účtů.
- **Show Coverage** – pokud má uživatel označen jeden účet a je-li tento účet pokryt rolemi v tabulce rolí, tato funkce ukáže, kterými rolemi je tento účet pokryt. Pokud daný účet není pokryt rolemi v tabulce rolí, vyznačí se problematické atributy červeně.

File menu

Menu **File** se nachází vlevo nahoře v horním okně. Dělí se na pět hlavních částí. Všechny tyto části jsou podrobně popsány níže, vyjma záložky exit, která jediná není dále rozbalovací, protože jen ukončí program.

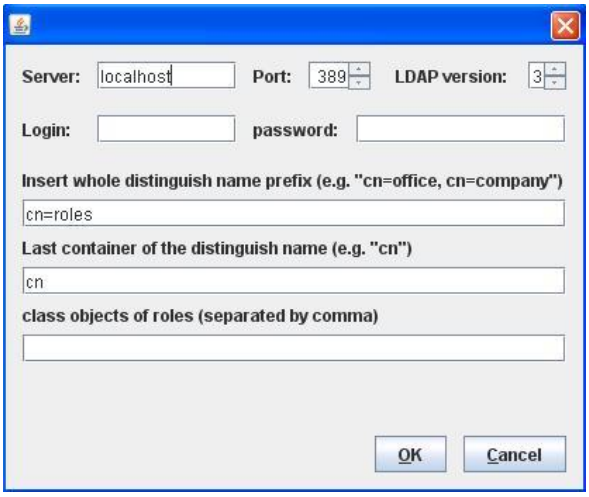


Příloha - Obrázek 6 menu File (zdroj vlastní)

- **Load Account -** uživatel má možnost načíst účty buď z LDIF souboru (alternativně lze použít také klávesovou zkratku CTRL+L) nebo z databáze LDAP. Pokud jsou v tabulce rolí uvedeny již nějaké role, budou tyto role po následujícím načtení účtů smazány. Doporučuje se proto umístěné role z tabulky rolí před změnou hlavních atributů exportovat.
    - Při načítání LDIF souboru je nezbytné, aby daný soubor přesně splňoval definici formátu LDIF, např. číslo verze na prvním řádku, dn u každého účtu atd. Formální specifikaci formátu LDIF je k nalezení v RFC2849.
    - Při načítání z databáze LDAP se objeví následující pole:

- **Server** – adresa serveru, na který si uživatel přeje role uložit.
- **Port** – port, přes který se lze připojit k danému serveru.
- **LDAP version** – verze LDAP na daném serveru.
- **Login a Password** – přihlašovací údaje na daný server.
- **Search base** – cesta, z které se mají role z databáze nahrát.
- **Filter expression** – filter se uplatní při načítání účtů.

- **Save roles** - Role lze ukládat následujícími způsoby buď do dvou souborových formátů CSV a LDIF, nebo na LDAP server. Zřejmou podmínkou pro funkci volby je existence role, kterou lze uložit. Pokud si uživatel vybere formáty CSV nebo LDIF, otevře se souborový manažer, pomocí něhož uživatel role uloží do odpovídajících formátů. Pro uložení do formátu CSV lze alternativně použít také klávesovou zkratku CTRL+S.



Příloha - Obrázek 7 LDAP export (zdroj vlastní)

Pokud si uživatel zvolí uložení rolí do LDAP databáze, objeví se okno viz. Obrázek 7.:

- **Server** – adresa serveru, na který si uživatel přeje role uložit.
- **Port** – port, přes který se lze připojit k danému serveru.
- **LDAP version** – verze LDAP na daném serveru.
- **Login a Password** – přihlašovací údaje na daný server.
- **Distinguish name prefix**– cesta, do které se mají role v databázi uložit.

- **Last container of the distinguish name** – atribut pro koncovou část distinguish name, který se bude rovnat jménu ukládané role, např.:
    - Je cíleno uložit dvě role, které mají atributy Telephone a AccountID, každá role má navíc kromě jiného i své jméno (name) - 1. name: Role1, Telephone: 123456, AccountID: 4576; 2. name: Role2, Telephone: 123789, AccountID: 6754. Distinguish name prefix je vyplněno na: cn=roles,DC=Mujldap, Last container of the distinguish name je r. V tomto případě se role uloží pod těmito distinguish name: 1. dn: r=Role1, cn=roles, DC=Mujldap; 2. dn: r=Role2, cn=roles, DC=Mujldap.
- **Class objects of roles** – objektové třídy, pod kterými si přeje uživatel dané role uložit například „first_level_roles“.

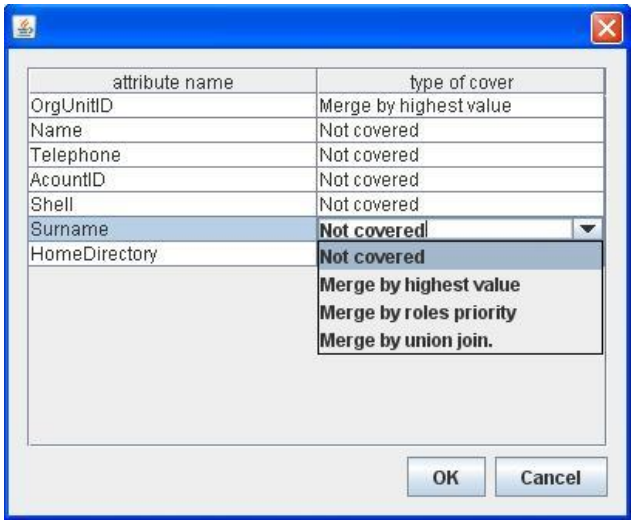
Uživatel si musí sám zkontrolovat, zda role splňují všechna kritéria objektových tříd, pod kterými si přeje role uložit. Navíc se objektové třídy definují i **Last container of the distinguish name.** Přeje-li si uživatel například uložit role s atributy Telephone a AccountID jako instance objektové třídy Role, musí objektová třída Role atributy Telephone a AccountID obsahovat a navíc nesmí obsahovat žádný další povinný atribut. Dále musí uživatel správně dle objektové třídy Role vyplnit **Last container of the distinguish name.**

Příloha - Obrázek 8 Nesplnění podmínek dané objektové třídy LDAP (zdroj vlastní)

- **Import roles -** Systém umožňuje uživateli importovat do programu Role modeler již dříve uložené role. Tato funkcionalita podporuje souborový formát LDIF. Podmínkou pro import rolí je určení hlavních atributů.
    - Importovat role lze pouze za předpokladu, jsou-li nastaveny hlavní atributy. Pokud importované role některý z nastavených hlavních atributů neobsahují nebo obsahují nekorektní hodnotu např. nečíselná hodnota u $higest - value$ atributu, dosadí se prázdná hodnota.

- **Choose -** Touto volbou si uživatel nastavuje záložku zobrazení atributů v tabulce účtů. Může si zvolit hlavní atributy - **choose main attributes**, lze též použít klávesovou zkratku CTRL+M nebo viditelné atributy - **choose attributes to view**, lze též použít klávesovou zkratku CTRL+W.

Při výběru nastavení hlavních atributů se ukáže následující okno.



Příloha - Obrázek 9 Hlavní atributy (zdroj vlastní)

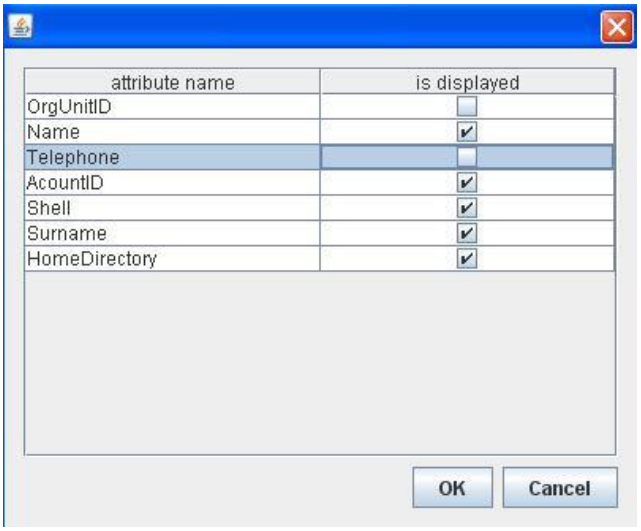
Kliknutím do sloupce **type of cover** se rozbalí u daného atributu roleta, kde si lze vybrat z následujících možností:

- **Not covered** - takový atribut se nebude pokrývat, nebude hlavním atributem.
- **Merge by highest value** - takový atribut bude typu highest-value.
- **Merge by priority** - takový atribut bude typu priority.
- **Merge by union join** - takový atribut bude typu multi-value.

Pokud jsou v tabulce rolí uvedeny již nějaké role, budou tyto role po změně hlavních atributů smazány. Doporučuje se proto umístěné role z tabulky rolí před změnou hlavních atributů exportovat. Pokud uživatel označí multi-value atribut jako typ highest-value nebo priority, systém se pokusí odfiltrovat účty, které mají v multi-value atributu více než jednu hodnotu. Pokud existují účty, které nemají některý z hlavních atributů, systém na to upozorní a tyto nevyhovující účty odfiltruje. Výpis těchto nevyhovujících účtu je proveden do souboru **Rolemodeler.log**.

Příloha - Obrázek 10 Varování před filtrací multi-value a nevyhovujících účtů (zdroj vlastní)

Záložka viditelných atributů (Obrázek 11) umožňuje zaškrtnutím pole **is displayed** nastavení viditelnosti daného atributu v tabulce **Accounts by atribute to view**.



Příloha - Obrázek 11 Viditelné atributy (zdroj vlastní)

Find roles – možnosti nastavení

Tlačítkem **Find roles** je zahájena hlavní funkcionalita aplikace, podmínkou pro hledání rolí je určení hlavních atributů. Aplikace umožňuje řešit dvě různé úlohy týkající se modelování rolí:

- Na základě zadání počtu rolí se systém pokusí nalézt požadovaný počet takových rolí, aby pokrývaly co nejvíce účtů. Pokud systém nalezne méně rolí, které pokrývají všechny účty, doplní je rolemi s prázdnými atributy – *maximalizační problém*. Uživatel nemůže zadat při *maximalizačním problému* vyhledání menšího počtu rolí, než je počet rolí předdefinovaných, ani méně rolí, než je počet hodnot případného fixovaného atributu.
- Při zadání, kolik procent účtů má být pokryto, se systém pokusí nalézt co nejmenší počet rolí, aby pokrýval zadaný počet účtů – *minimalizační problém*

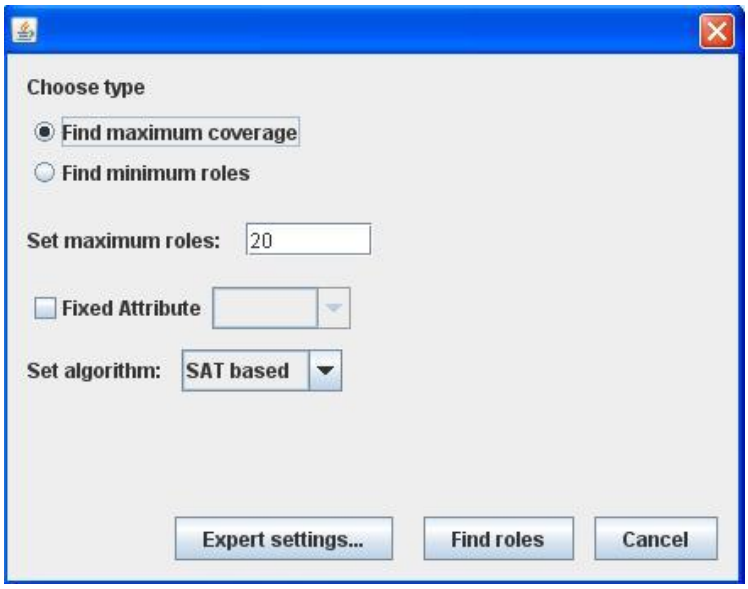
Uživatel má ještě možnost tzv. *fixovat atribut*. To znamená, že se určitě vytvoří role, které budou pokrývat všechny hodnoty tohoto atributu (všechny hodnoty, které se vyskytují u načtených neodfiltrovaných účtů). Navíc má uživatel ještě možnost stanovit si v hlavním okně předdefinované role, které budou určitě v množině algoritmem „nalezených rolí".

Podstatné vlastnosti algoritmů:

- Řešení *maximalizačních úloh* trvá výrazněji kratší dobu než řešení *minimalizačních úloh.*
- Někdy algoritmus poměrně rychle najde své řešení, ale potom se ho pokouší ještě velmi dlouho zlepšovat.
- Výrazněji delší délka řešení maximalizačního problému, pokud neexistuje 100% zadaným počtem rolí.

Z první vlastnosti plyne, že pokud si uživatel přeje velmi přesné výsledky u *minimalizačního problému*, je doporučeno danou úlohu nejprve pokusit vyřešit pomocí některého z algoritmů jako *minimalizační problém* a potom zkusit *maximalizační úlohu* na nižší počet rolí.

Po kliknutí na tlačítko Find roles se otevře okno s následujícími ovládacími prvky.



Příloha - Obrázek 12 Find Roles (zdroj vlastní)

U **Choose type** lze vybrat úlohu, kterou bude **Role modeler** řešit:

- **Find maximum coverage** – *maximalizační problém*, počet rolí se poté zadává do políčka **Set maximum roles,** logicky zadat jen kladné celé číslo.
- **Find minimum roles** – *minimalizační problém*, procentuální pokrytí účtu se poté zadává do políčka **Set minimum coverage (%)**, opět lze zadat jen kladné celé číslo v rozmezí 1-100.

Pokud uživatel zaškrtne pole **Fixed Attribute**, může napravo od něj rozbalit roletu s nabídkou hlavních atributů k *fixaci.*

Výběr algoritmů řešících úlohu probíhá v roletě **Set algorithm**, pod ní lze pomocí tlačítka **Expert settings** vyvolat pokročilejší nastavení daného algoritmu.

Parametry algoritmu SAT-based

Jádrem tohoto algoritmu je řešič problému SAT. Algoritmus převede předepsaný problém na formuli, kterou poté předá řešiči.

Expert settings u Sat-based algoritmu obsahuje jen jedno pokročilé nastavení a to **Maximum interval for the internal solver**. Algoritmus si dělí výpočet na malé pod úlohy a toto nastavení udává maximální dobu výpočtu jednotlivých pod úlohy. Čím bude tento interval větší, tím lepší bude výsledek algoritmu, ale o to horší bude interakce programu s uživatelem. Tlačítko Stop, sloužící na zastavení běhu algoritmu, může zareagovat v nejhorším případě až po uplynutí zadaného intervalu a číselný údaj ukazující pokrok výpočtu, napravo dole v hlavním okně, se mění také v závislosti na zadaném intervalu.

Pokud si uživatel před spuštěním algoritmu nevyvolá okno **Expert settings**, je pro **Maximum interval for the internal solver** použita přednastavená hodnota, která se odvíjí od počtu účtů. Tyto hodnoty jsou vybrány více s ohledem na přesnost řešení v neprospěch interakce programu s uživatelem.

Problémem SAT-based algoritmu je jeho paměťová náročnost. Řešiči se musí předat formule kompletně popisující problém k vyřešení a ta u většího počtu účtů nebo hledaných rolí může enormně narůst.

Příloha - Obrázek 13 Upozornění na nedostatek paměti (zdroj vlastní)

Parametry algoritmu Hill Climb

Algoritmus Hill Climb je prohledávání stavového prostoru s cílem neustálého zlepšování výsledku. Lze si to představit jako horolezce, který se snaží neustále lézt někam nahoru, dokud to jde. Je zřejmé, že čím více horolezců je postupně vyzkoušeno v rozlehlých horách, tím lepší bude celkový výsledek a o to déle také polezou. Tento algoritmus byl během testů nejrychlejší a měl nejlepší výsledky, jeho problémem je jen výraznější zpomalení při velkém počtu hledaných rolí. Také může chvíli trvat, než je zaznamenán vpravo dole v hlavním okně první pokrok.

Okno **Expert settings** se u algoritmu Hill Climb vyplňuje jen při *maximalizačním problému*. *Minimalizační problém* má rozsáhlé optimalizace, kvůli kterým se nevyplatí nastavovat např. počet horolezců a jeho expertní nastavení neexistuje.

Maximalizační problém obsahuje tato nastavení:

- **Number of climbers** - počet horolezců je přímo úměrný výpočetní době. Horolezci „lezou“ postupně po jednom a zkouší nalézt co nejlepší řešení, pokud je zapnuta heuristika, toto číslo obsahuje i první tři heuristické horolezce. Přednastavená hodnota je pět, testy bylo ověřeno, že už tento počet horolezců generuje velmi dobré výsledky.
- **Use heuristic** - heuristikou jsou vlastně tři heurističtí horolezci, kteří se snaží nasměrovat výpočet správným směrem, je to deterministický proces, vypnutí může přijít vhod při hledání více různých řešení nebo generování možných návrhů.
- **Stop after cover #accounts** - uživatel má tímto nastavením možnost určit hranici počtu pokrytých účtů, při které se algoritmus automaticky zastaví, je to možné nastavit jen při *maximalizačním problému*.

Parametry Genetického algoritmu

Genetický algoritmus je inspirovaný přirozenou evolucí. Aby tento postup našel dobré řešení, simuluje populaci možných řešení a pamatuje si nejlepší řešení. Běh algoritmu je rozdělen na generace. V každé generaci je určitý počet jedinců odstraněn. Tito odstranění jedinci jsou nahrazeni novými pomocí různých genetických metod. Genetický algoritmus využívá i možnosti malého horolezce, který se snaží genetické metody nasměrovat.

Tento algoritmus je značně založený na náhodě a mívá kolísavé výsledky. Při testování měl nejisté výsledky zejména při minimalizačním problému.

Genetický algoritmus má mnoho expertních nastavení, nejprve jsou jednotlivá pole popsána a potom je jejich použití ilustrováno na názorném příkladu.

- **Maximum population size** - maximální počet řešení v populaci, tzn. počet, na který se populace doplní při každém generování.
- **Mutation size** - počet řešení vytvořených metodou mutace při generování v jedné generaci.
- **Merge size** - počet řešení vytvořených metodou spojení při generování v jedné generaci.
- **Maximum count of bad instances** - maximální počet náhodně vybraných řešení při selekci, maximální z toho důvodu, že tento počet se vybere náhodně ze všech řešení, i když by třeba některá takto vybraná řešení stejně byla ponechána, protože spadají mezi vhodná.
- **Hill Climb size** - počet řešení vytvořených metodou horolezce při generování v jedné generaci.
- **Hill Climb ticks** - počet hodnot, které jednoduchý horolezec vyzkouší, aby našel lepší řešení.
- **Maximum count of the same generation** - parametr ukončení algoritmu, po kolika generacích bude algoritmus ukončen, pokud se během nich neobjevilo žádné lepší řešení.

Příklad:

Velikost populace 1000, velikost mutace 400, velikost spojení 100, velikost horolezce 20 a maximum nevhodných řešení 50.

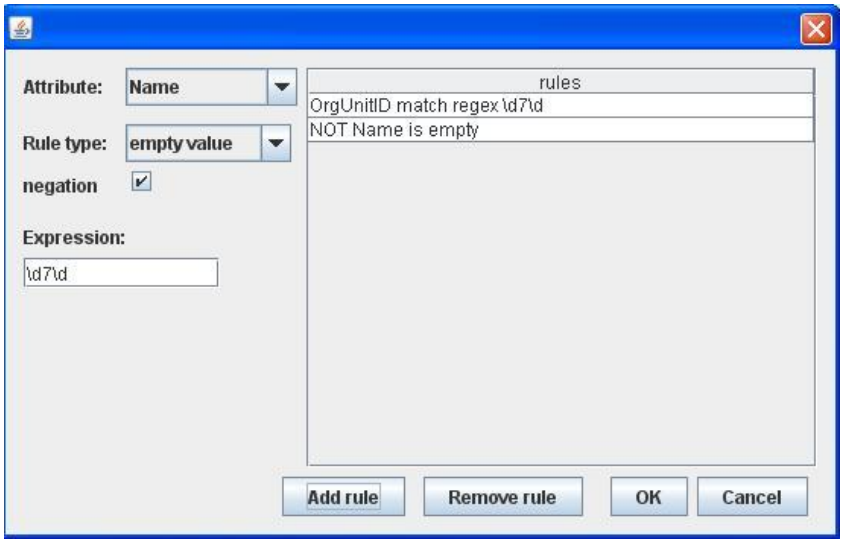
Pří každé selekci bude zachováno maximálně 1000 – 400 – 100 – 20 = 480 řešení. Z tohoto počtu řešení bude 480 – 50 = 430 nejlepších řešení v populaci a 50 náhodně vybraných řešení z celé populace. Tedy po selekci zůstane v populaci počet řešení mezi 430 a 480, Podle toho, kolik z těchto 50 náhodně vybraných už je obsaženo v 430 nejlepších řešeních. Například zbylo 460 jedinců.

Při další generaci bude naplánováno vytvoření 20 nových jedinců horolezcem (což může selhat), 100 jedinců spojením a zbytek pomocí mutace. Tedy volné místo je 1000 – 460 = 540, 540 – 20 – 100 = 420 nových jedinců bude vytvořeno pomocí mutace. Je to o 20 jedinců více než zadaný počet velikosti mutace, je to tolik řešení, kolik jich bylo při selekci „vybráno dvakrát".

Pomocný algoritmus Brutal Force

Brutal Force je pracovní název pro jednoduchý algoritmus, který hrubě vyzkouší všechny možnosti problému. Implementován je z důvodu testování a pro vytvoření přehledu o výkonnosti ostatních algoritmů. Pro běžného uživatele je v podstatě nepoužitelný.

Filter



Příloha - Obrázek 14 Filter (zdroj vlastní)

V této funkcionalitě má uživatel možnost vytvořit si pravidla, která budou muset splňovat načtené účty. Účet je odfiltrován, pokud nesplňuje kterékoli z definovaných pravidel.

Ve filtrovacím okně jsou na levé straně ovládací prvky pro nastavení pravidel a na pravé straně tabulka **rules** se samotnými pravidly. Pod tabulkou jsou tlačítka **Add rule** a **Delete rule.** První zmiňované vytvoří právě nastavené pravidlo a druhé odstraní označené

pravidlo/označená pravidla z tabulky **rules.** Jednotlivé ovládací prvky k nastavování pravidel jsou popsány v pořadí odshora dolů.

V roletě u položky **Atribute,** lze nastavit atribut, kterého se bude dané pravidlo týkat. V roletě u položky **Filter type** si uživatel může vybrat jedno z následujících pravidel:

- **Empty value** – budou odfiltrovány všechny účty, které neobsahují v daném atributu prázdnou hodnotu, toto pravidlo je zajímavé spíše v kombinaci s negací pomocí pole **negation.**
- **Equal to string** – budou odfiltrovány všechny účty, které nemají v daném atributu hodnotu rovnu specifikované hodnotě v poli **Expression.**
- **Match to regex** - budou odfiltrovány všechny účty, které nemají v daném atributu hodnotu odpovídající specifikovanému regulárnímu výrazu v poli **Expression.**

Všechna tato pravidla lze ještě navíc znegovat zaškrtnutím políčka **negation**.

Pokud uživatel zadá pravidlo týkající se *multi-value* atributu, účet toto pravidlo splňuje, když ho splňuje jakákoli z hodnot daného *multi-value* atributu.

Regulární výrazy

Tato podkapitola popisuje základy používání regulárních výrazů, které jsou pro tvorbu naprosté většiny filtrů postačující.

Regulární výraz je řetězec znaků, který zastupuje více jiných řetězců. Většina znaků zastupuje sebe sama například všechny alfanumerické znaky, jsou však i znaky, které mají speciální vlastnosti – tzv. *metaznaky*. Je jich celkem 11 a jsou jimi: **(\ ( ) * + . $ ^ | ?**

Obrácené lomítko **** má tu speciální vlastnost, že mění význam znaku před ním. Některým jiným znakům než metaznakům umožňuje speciální vlastnosti a naopak pokud uživatel potřebuje, aby *metaznak* zastupoval sám sebe a neměl speciální vlastnost, uvede ho znakem **\.**

Například znak **+** znamená libovolný, nenulový počet opakování předchozího znaku. Regulární výraz **1+1=2** tedy zastupuje řetězce **111=2**, **1111=2**, **11111=2** atd. Regulární výraz **1\+1=2** už zastupuje opravdu jen řetězec **1+1=2**, protože obrácené lomítko zruší speciální vlastnost *metaznaku* **+**.

Znak	Speciální vlastnost
X?	znak X s jedním nebo žádným počtem opakování
X*	znak X s libovolným počtem opakování (i nulovým počtem opakování)
X+	znak X s libovolným nenulovým počtem opakování
X{n}	znak X opakovaný přesně n krát
X{n,}	znak X opakovaný minimálně n krát
X{n,m}	znak X opakovaný n až m krát
(X|Y)	buď znak X nebo znak Y
(abc)	znak a, b, nebo c
(^abc)	jakýkoli znak mimo znaků abc (negace)
(a-m)	jakýkoli znak z abecedy od a po m bez diakritiky
(a-zA-Z)	jakýkoli znak z abecedy a-z sjednoceno s A-Z (všechna písmena bez diakritiky)
(abcg-j)	jakýkoli znak ze znaků abcghij
.	zastupuje kterýkoli jeden znak
\d	jakýkoli číselný znak
\D	jakýkoli nečíselný znak
\s	jakýkoli bílý znak (mezera, tabulátor apod.)
\S	jakýkoli nebílý znak
\w	jakýkoli alfanumerický znak
\W	jakýkoli nealfanumerický znak

Příloha - Tabulka 1 Znaky s vybranými speciálními vlastnostmi

Ukázky použití regulárních výrazů:

Pe(t|p)a popisuje řetězce „Peta“ a „Pepa“.

Ba*f popisuje řetězce „Bf“, „Baf“, „Baaf“, „Baaaf“ atd.

\d{3} \d{2} popisuje formát PSČ – posloupnost tří číslic, mezeru a dvě číslice.

<(^>)*> popisuje tag v jazyce HTML, libovolný text uzavřený mezi špičaté závorky (mezi špičatými závorkami můžou být jakékoli znaky kromě „>“).
(0-9a-fA-F)+(, ?(0-9a-fA-F)+)* popisuje seznam hexadecimálních čísel, oddělených čárkami a nepovinnými mezerami, např. „8A,9C,4f“ nebo „fa, BC, d8“.

Příloha 2 – Karty procesů neziskových organizací

Proces příprava podkladů

Cíl procesů	Aktivita, při které dochází k zpracování projektových žádostí.
Popis procesu	V rámci procesu se uskutečňuje příprava žádosti o dotační projekt.
Přidaná hodnota	Zajištění plánovaných aktivit nadace a činností v oblasti osvěty veřejného zdraví.
Zákazník procesu	ORGANIZACE_1_C
Vlastník procesu – odpovědnost zaměstnanců	Nikola_34, Alena_23, Radka_2
Spolupráce v rámci organizace	Administrativní pracovnice + manažer dotačního projektu
Klíčové vstupy	Požadavek na návrh projektu
Klíčové výstupy	Žádost o projekt
Podpůrné procesy	2.1.1. Úvodní fáze projektu
Posloupnost činností	2.1.1. Úvodní fáze projektu Metodika zpracovaní pokladů - vytvoření seznamu a šablon, sběr a sumarizace projektových záměrů a jejich předložení ke schválení, dále, sběr veškerých potřebných podkladů dle podmínek dotace, odeslání schvalování projektu.
Časové trvání průběhu činnosti a režim aktivity	Délka realizace: 3 měsíce Režim aktivity: nárazový
Podpora ICT	e-mailová komunikace, sdílené úložiště

Proces definice pravidel

Cíl procesů	Aktivita, při které dochází k definici pravidel projektů.
Popis procesu	V rámci procesu se uskutečňuje definice a aktualizace pravidel realizace projektů spolu s definicí modelu financování.
Přidaná hodnota	Zajištění plánovaných aktivit nadace a činností v oblasti osvěty veřejného zdraví.
Zákazník procesu	ORGANIZACE_1_C
Vlastník procesu – odpovědnost zaměstnanců	Nikola_34, Alena_23, Radka_2
Spolupráce v rámci organizace	Administrativní pracovnice + manažer dotačního projektu
Klíčové vstupy	Požadavek definice pravidel
Klíčové výstupy	Šablona projektu, soubor pravidel, smlouvy o realizaci projektu
Podpůrné procesy	2.2.1. Plnění definice pravidel
Posloupnost činností	2.2.1. Plnění definice pravidel Prezentace a objasnění pravidel spolupracujícím organizacím, vytvoření, aktualizace, předání a objasnění využívání šablony projektového záměru, předání a objasnění využívání šablon pro potřebné podklady dle podmínek dotace vedoucím projektů. Návrh realizace projektu, připomínkování, předložení spolupracujícím organizacím, kteří ho realizují, realizace administrativy.
Časové trvání průběhu činnosti a režim aktivity	Délka realizace: 2 měsíce Režim aktivity: nárazový
Podpora ICT	mailová komunikace, interní a sdílené úložiště

Proces monitoring

Cíl procesů	Aktivita, při které dochází ke kontrole plnění cílů projektu.
Popis procesu	V rámci procesu se uskutečňuje monitoring projektu.
Přidaná hodnota	Zajištění plánovaných aktivit nadace a činností v oblasti osvěty veřejného zdraví.
Zákazník procesu	ORGANIZACE_1_C
Vlastník procesu – odpovědnost zaměstnanců	Nikola_34, Alena_23, Radka_2, Lubomir_1
Spolupráce v rámci organizace	Administrativní pracovnice + manažer dotačního projektu + manažer nadace
Klíčové vstupy	Data
Klíčové výstupy	Dílčí zprávy z jednotlivých etap
Podpůrné procesy	2.3.1. Reporting ministerstvo 2.3.2 Reporting nadace
Posloupnost činností	2.3.1. Reporting ministerstvo Realizace průběžné pololetní zprávy, průběžné etapové a monitorovací zprávy, kontrola kompletnosti podkladů, příprava zpráv, odsouhlasení manažerem, zajištění podpisu zprávy a odeslání, kontrola kompletnosti dokladů, vyplnění zprávy, podání a podpis zprávy, dodání podkladů za předchozí kvartál do 3. dne následujícího měsíce. 2.3.2. Reporting nadace Zprávy z výsledků projektů + poklady z projektů, kontrola plnění monitorovacích, kontrola harmonogramu, kontrola výkazů práce.
Časové trvání průběhu činnosti a režim aktivity	Délka realizace: 2 měsíce Režim aktivity: nárazový
Podpora ICT	e-mailová komunikace, interní a sdílené úložiště, webové stránky vč. intranetu

Financování

Cíl procesů	Aktivita, při které dochází k proplacení určité etapy projektu.
Popis procesu	V rámci procesu se uskutečňuje realizace žádosti o platbu.
Přidaná hodnota	Zajištění plánovaných aktivit nadace a činností v oblasti osvěty veřejného zdraví.
Zákazník procesu	ORGANIZACE_1_C
Vlastník procesu – odpovědnost zaměstnanců	Nikola_34, Alena_23, Radka_2
Spolupráce v rámci organizace	Administrativní pracovnice + manažer dotačního projektu
Klíčové vstupy	Data - Financování projektu.
Klíčové výstupy	Realizace plateb.
Podpůrné procesy	2.4.1. Plnění žádosti o platbu.
Posloupnost činností	2.4.1. Plnění žádosti o platbu. Součinnost při přípravě podkladů pro žádost o platbu, příprava do složek, kopírování dokladů, podání a podpis zprávy, tištěné dokumenty na ministerstvo. Žádost o platbu za dílčí etapu podána s podpisem, doručení zpracované žádosti (složky) na ministerstvo projektovému manažerovi, po kompletním interním schválení je žádost o platbu předána na ministerstvo, po úspěšném interním schválení je žádost o platbu proplacena na bankovní účet nadace. Je nutné doplnit účetní doklady, úhrady, bankovní výpisy, úhradové doklady (pokladna), doložit přílohy (sken) – všechny účetní doklady (účetní a úhradové), objednávky, pracovní smlouvy (dodatky i výpovědi), popř. další smlouvy týkající se projektu, dodací listy spolu s inventárními kartami (HIM), rozpis mzdových nákladů, zprávu realizované činnosti v sídle a provozovnách, dílčí výzkumné zprávy k jednotlivým výzkumným projektům, zprávy z jednotlivých akcí a seminářů. Rozpis mzdových nákladů a jednotlivé zprávy musí být parafovány statutární osobou. Kopie účetních dokladů a smluv spolu s parafovanými originály rozpisu mzdových nákladů a zpráv jsou zpracovány do složky. Jednotlivé dokumenty jsou označeny: „účetní doklad č.“, „úhradový doklad č.“, „příloha č.“.

Časové trvání průběhu činnosti a režim aktivity	Délka realizace: 3 měsíce Režim aktivity: nárazový
Podpora ICT	e-mailová komunikace, úložiště interní.

Realizace projektu

Cíl procesů	Aktivita, při které dochází k realizaci vytyčených cílů.
Popis procesu	V rámci procesu se uskutečňuje realizace projektu.
Přidaná hodnota	Zajištění plánovaných aktivit nadace a činností v oblasti osvěty veřejného zdraví.
Zákazník procesu	ORGANIZACE_1_C
Vlastník procesu – odpovědnost zaměstnanců	Nikola_34, Alena_23, Radka_2 + realizační tým
Spolupráce v rámci organizace	Administrativní pracovnice + manažer dotačního projektu + realizační tým
Klíčové vstupy	Data
Klíčové výstupy	Činnosti spojené s realizací projektu
Podpůrné procesy	2.5.1. Realizační činnosti
Posloupnost činností	Na základě definice a popisu projektu probíhá jeho realizace. Projekt by měl být plněn dle navrhovaného harmonogramu s přihlédnutím na rozpočet projektu. Financování projektu závisí na přidělené dotaci a na realizaci smluv od sponzorů. O zajištění personálního obsazení se stará vedoucí projektu stejně jako o technickou infrastrukturu. Projekt musí být plněn dle předem stanovených akceptačních kritérií. Výstupem projektu jsou průběžné a monitorovací zprávy předkládané ministerstvu a také zprávy překládané v rámci nadace a spolupracujícím subjektům. Nadace jako zastřešující organizace má za povinnosti provádět monitoring svých projektů.
Časové trvání průběhu činnosti a režim aktivity	Délka realizace: 2-3 roky Režim aktivity: nárazový
Podpora ICT	Interní a sdílené úložiště, webové stránky včetně intranetu

Archivace

Cíl procesů	Aktivita, při které dochází ke sběru a archivaci výstupů projektů.
Popis procesu	V rámci procesu se uskutečňují činnosti pro zachování dokumentů spojených s projekty.
Přidaná hodnota	Zajištění plánovaných aktivit nadace a činností v oblasti osvěty veřejného zdraví.
Zákazník procesu	ORGANIZACE_1_C
Vlastník procesu – odpovědnost zaměstnanců	Nikola_34, Alena_23, Radka_2
Spolupráce v rámci organizace	Administrativní pracovnice + manažer dotačního projektu
Klíčové vstupy	Dokumenty
Klíčové výstupy	Dokumenty
Podpůrné procesy	2.6.1. Plnění činnosti archivace
Posloupnost činností	- Archiv: pracovní smlouvy, nájemní smlouvy, smlouva o účetnictví, o bankovním účtu, kopie smluv s dodavateli.
Časové trvání průběhu činnosti a režim aktivity	Délka realizace: 2 měsíce Režim aktivity: nárazový
Podpora ICT	Interní úložiště