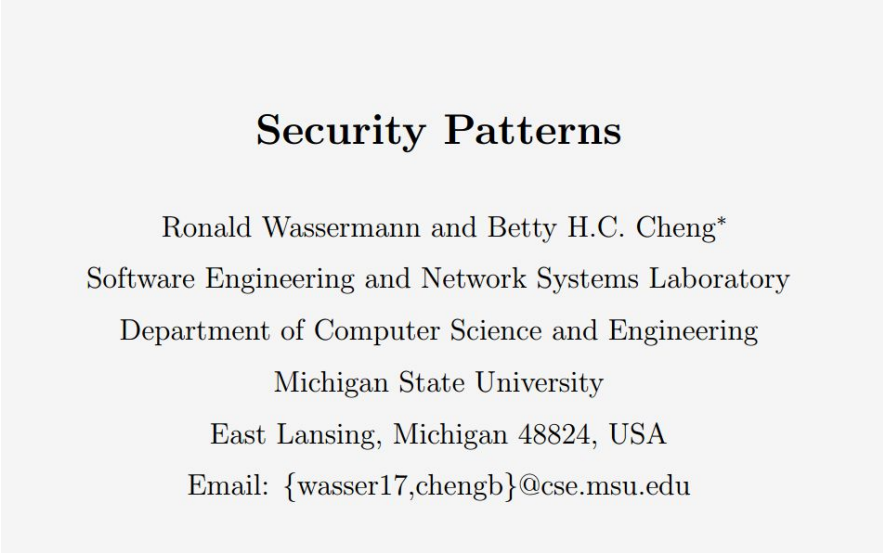
Security patterns

PV167 ; D. Veselý ; O. Smětáк

Zdroj pro následující část

- paper **Security Patterns**
- Ronald Wassermann and Betty H.C. Cheng
- https://www.cse.msu.edu/~cse870/Materials/security-patterns.pdf
- 30+ citací



Security Patterns

Ronald Wassermann and Betty H.C. Cheng*
Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824, USA
Email: {wasser17,chengb}@cse.msu.edu

Full View With Errors vs Limited View

- **Full View With Errors**
- uživatel vidí všechny ovládací prvky
- pokud provede nepovolenou akci
  - zastaví ho chybová hláška

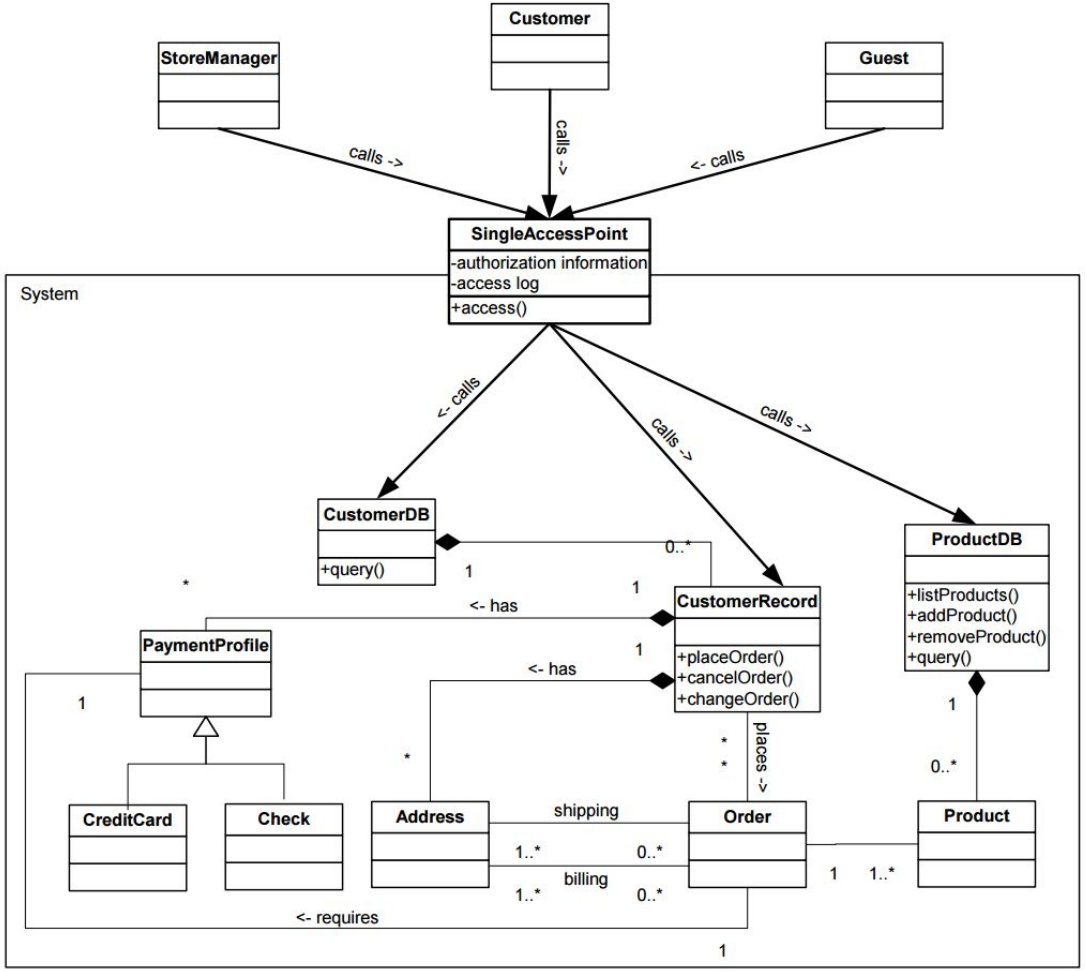
- podvědomě uživatel zná všechny možnosti systému (někdy může být výhoda)

- **Limited View**
- uživatel vidí vybrané ovládací prvky
  - a to ty, které může skutečně použít

- jednodušší pro uživatele

Single Access Point

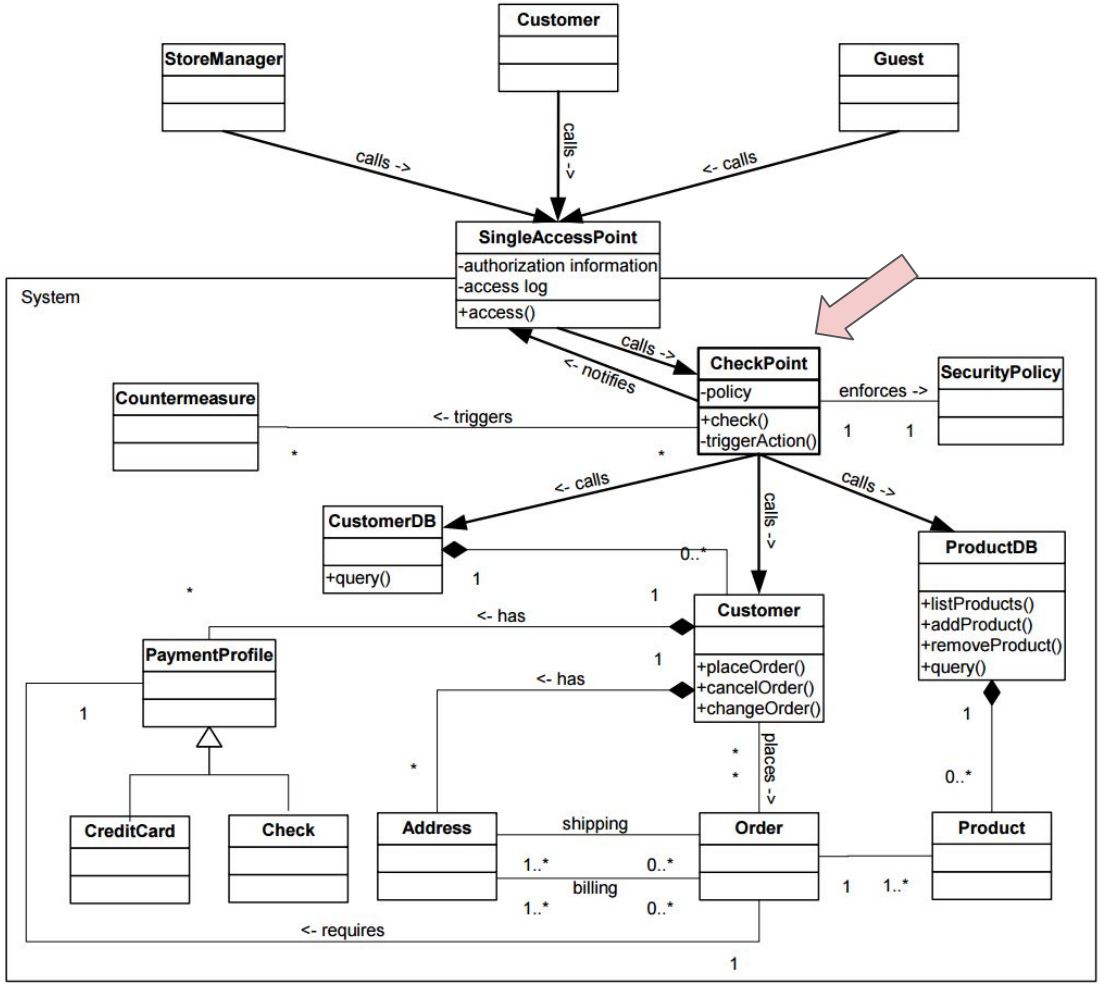
- jedno rozhraní (interface) pro všechnu komunikaci s entitami systému
- lepší kontrola a monitoring



zjednodušený e-shop

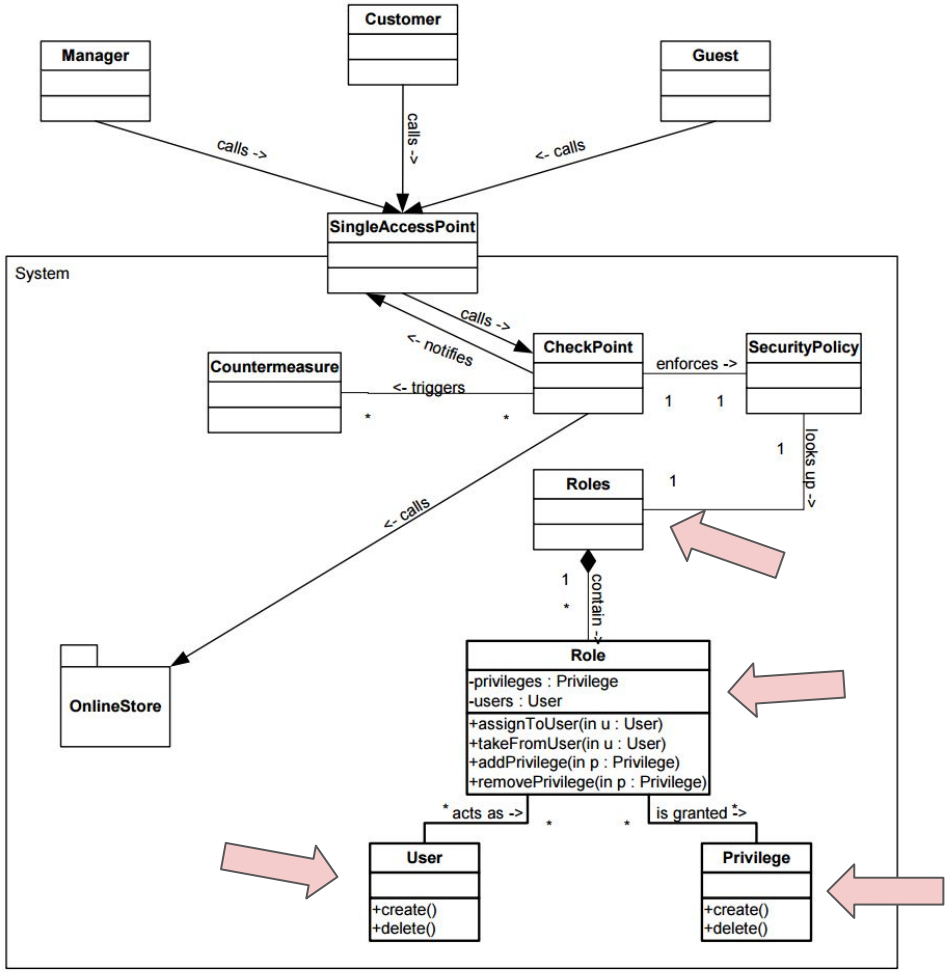
Check Point

- struktura kontrolující příchozí požadavky
- v případě narušení provede vhodná opatření (např. výjimka, záznam do logu)
- vhodné kombinovat s **Single Access Point**, zajistí kontrolu všech požadavků


StoreManager
Customer
Guest
calls ->
calls ->
<- calls
SingleAccessPoint
-authorization information
-access log
+access()
System
calls ->
<- notifies
CheckPoint
-policy
+check()
-triggerAction()
enforces ->
SecurityPolicy
Countermeasure
<- triggers
1
1
*
*
<- calls
calls ->
calls ->
CustomerDB
+query()
0..*
1
ProductDB
+listProducts()
+addProduct()
+removeProduct()
+query()
*
1
<- has
Customer
+placeOrder()
+cancelOrder()
+changeOrder()
PaymentProfile
1
<- has
1
*
*
*
places ->
1
0..*
CreditCard
Check
Address
shipping
Order
Product
1..*
0..*
billing
1..*
0..*
1
1..*
<- requires
1

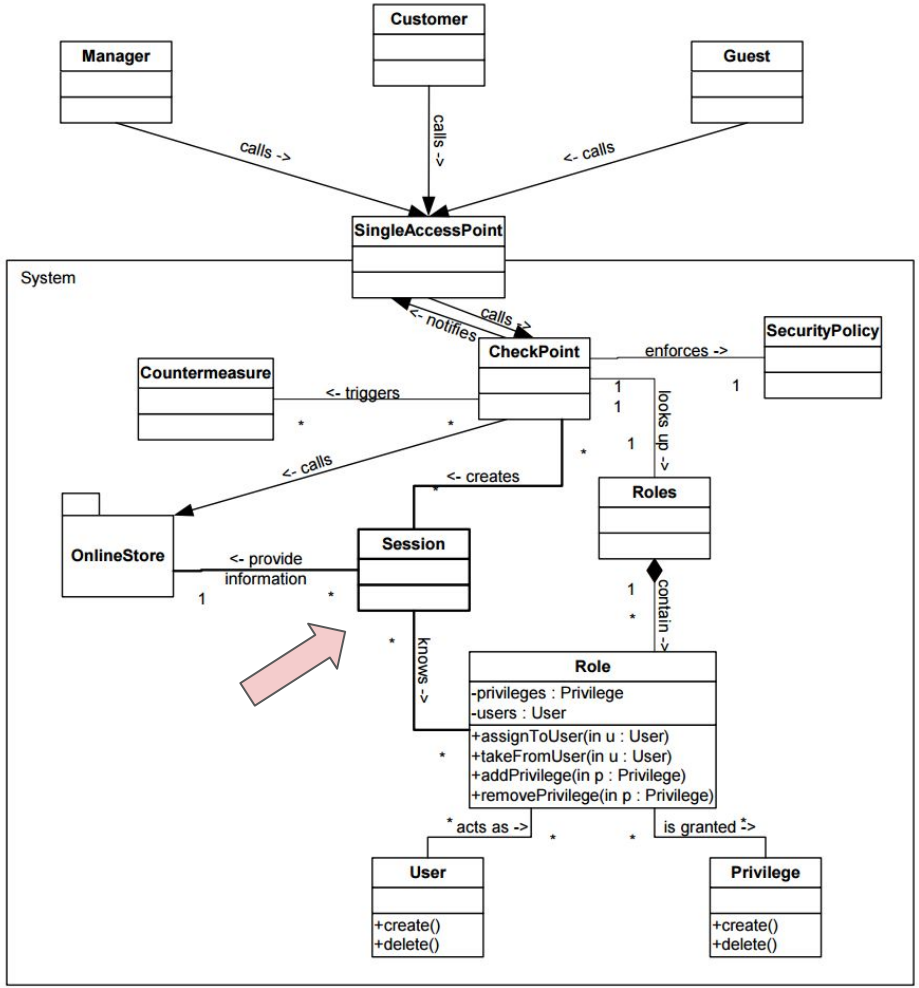
Roles

- pro lepší udržitelnost a snadnější administraci práv v systému
- abstrahuje od práv pro konkrétního uživatele
- role může být komplexní struktura práv
  - rolí je však v systému typicky výrazně méně než uživatelů


Manager
Customer
Guest
calls ->
calls ->
<- calls
SingleAccessPoint
System
calls ->
<- notifies
CheckPoint
enforces ->
SecurityPolicy
Countermeasure
<- triggers
1
1
looks up ->
Roles
<- calls
contain ->
OnlineStore
Role
-privileges : Privilege
-users : User
+assignToUser(in u : User)
+takeFromUser(in u : User)
+addPrivilege(in p : Privilege)
+removePrivilege(in p : Privilege)
acts as ->
is granted ->
User
+create()
+delete()
Privilege
+create()
+delete()

Session

- chceme pracovat s rozdílnými částmi systému
- a přitom chceme globálně přistupovat k informacím o uživateli
- typické použití Session je v případě HTTP, který je bezestavový
- Session vytváříme po autentizaci (přihlášení) uživatele

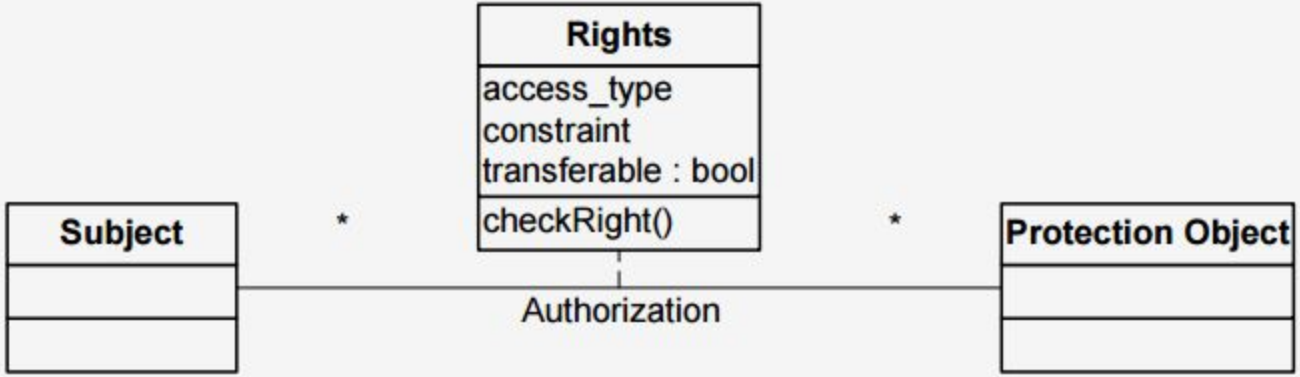

Manager
Customer
Guest
calls ->
calls ->
<- calls
SingleAccessPoint
System
<- notifies
calls ->
CheckPoint
enforces ->
SecurityPolicy
Countermeasure
<- triggers
looks up ->
<- calls
<- creates
Roles
OnlineStore
<- provide information
Session
contain ->
knows ->
Role
-privileges : Privilege
-users : User
+assignToUser(in u : User)
+takeFromUser(in u : User)
+addPrivilege(in p : Privilege)
+removePrivilege(in p : Privilege)
acts as ->
is granted ->
User
+create()
+delete()
Privilege
+create()
+delete()

Authorization

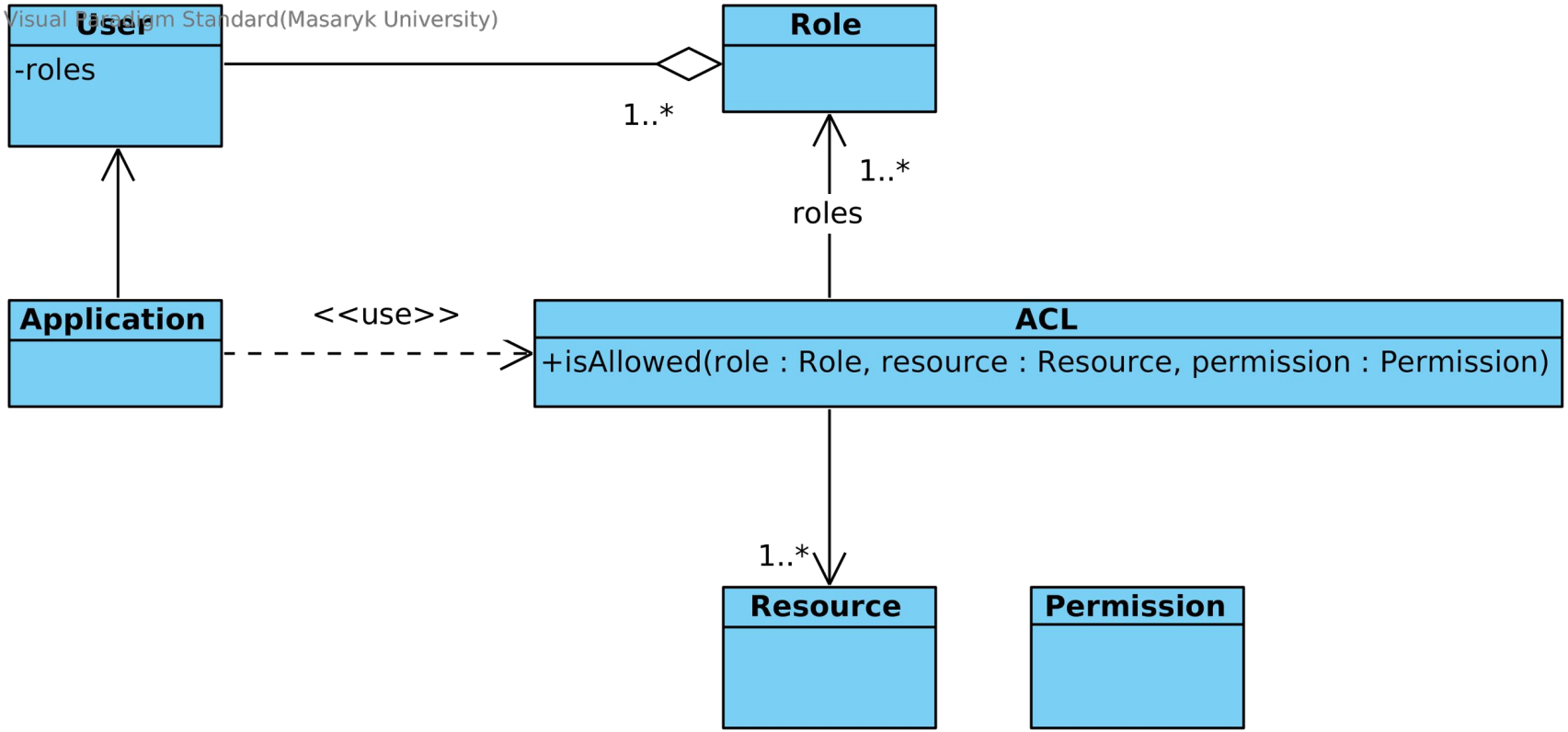
- usnadnění řízení přístupu k jednotlivým entitám systému
- **autorizace** je proces povolující uživateli určité akce
- snadná administrace např. díky ACL (access control list)
  - = tabulka říkající, **co kdo může** a potažmo nemůže

Příklad, použití v kódu (po implementaci):

```
// může guest prohlížet články?
echo $acl->isAllowed('guest', 'article', 'view'); // TRUE
// může guest editovat články?
echo $acl->isAllowed('guest', 'article', 'edit'); // FALSE
// může guest hlasovat v anketách?
echo $acl->isAllowed('guest', 'poll', 'vote'); // TRUE
// může guest komentovat?
echo $acl->isAllowed('guest', 'comment', 'add'); // FALSE
```


Rights
access_type
constraint
transferable : bool
checkRight()
Subject
*
*
Protection Object
Authorization

RBAC


Visual Paradigm Standard(Masaryk University)
User
-roles
Role
1..*
1..*
roles
Application
<<use>>
ACL
+isAllowed(role : Role, resource : Resource, permission : Permission)
1..*
Resource
Permission

Nedostatky RBAC

Představený způsob ACL se nazývá Role-based access control (RBAC)

RBAC je dobrý pro obecné vymezení oprávnění typu “Má uživatel v roli student přístup ke čtení zpráv?”

Co již však ACL neřeší je následné omezení typu “Ano uživatel v roli student má přístup ke čtení správ, ale pouze zpráv jemu určených.”

Obecné řešení tohoto problému je pomocí Attribute-based access control (ABAC)

Attribute-based access control (ABAC)

Narozdíl od RBAC vyhodnocuje přístupové oprávnění na základě několika různých atributů (například příjemce zprávy)

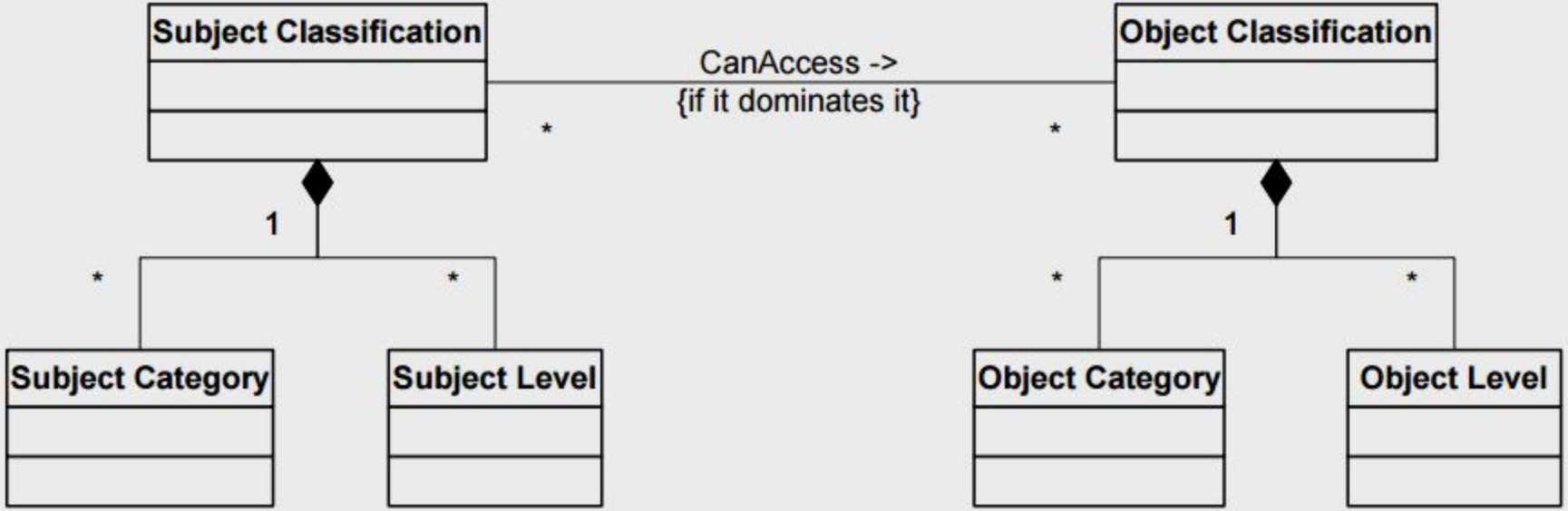
ABAC je lze implementovat pomocí tzv. XACML standard, pomocí kterého lze nadefinovat a řídit oprávnění přístupu

ABAC je často implementováno až na databázové vrstvě a nazývá Row-Level Security (RLS)

Základní operace RLS jsou blokace (nedovolí úpravu záznamu) a filtrace (omezí získávaná data jen na ta, ke kterým je povolený přístup pomocí tzv. **policies**)

Multilevel Security

- řízení přístupu v systému s rozdílnými úrovněmi přístupu
- např. podle NBÚ: přísně tajné > tajné > důvěrné > vyhrazené
- implementací je Bell–LaPadula model použitý v USA (Ministerstvo obrany)
  - top secret > secret > confidential > unclassified
  - spolehá na read down, write up property
  - **nemůžu číst** dokumenty s vyšším prověrkou, než mám
  - **můžu vytvářet** dokumenty s vyšší prověrkou, než mám
  - nasazuje se, když chceme zabránit úniku tajných informací (confidentiality)
- opakem Bell–LaPadula je model Biba
  - read up, write down property
  - **můžu číst** dokumenty s vyšším prověrkou, než mám
  - **nemůžu vytvářet** dokumenty s vyšší prověrkou, než mám
  - nasazuje se, když chceme zabránit modifikaci tajných informací (data integrity)


Subject Classification
CanAccess ->
{if it dominates it}
Object Classification
*
*
1
1
*
*
*
*
Subject Category
Subject Level
Object Category
Object Level